EasyIO Sedona kits
User Guide

Version 1.2
26th Sept 2012

Document Change Log

7th July 2011
Document created.

18th Oct 2011
Added MathConversion.kit

20th Oct 2011
Grammar correction for EasyioSox.kit
Minor changes

25th Oct 2011
Updated Pulse Accumulator details
Updated DEM5 wiring diagram

28th Oct 2011
Updated EasySox kit

31th Oct 2011
Updated EasyioTempTable kit. 14 Objects added. Version 1.0.45.22
Added new kit easyioEnergy.

1st March 2012
Updated EasyioTempTable kit. 2 Objects added. Version 1.0.45.23

Updated easyioDNS kit version 1.0.45.2
Updated easyioEmail.kit version to 1.0.45.1
Updated easyioEnergy kit version 1.0.45.1
Updated easyioPersistanceControl kit version 1.0.45.22
Updated easyioSchedule kit version 1.0.45.2 (Bug fixes)

Added easyioFGLcd.kit. Version 1.0.45.1

26th September 2012

Updated easyioP2P.kit version 1.0.45.1 – include client monitor status, (Boolean output)
Added easyioLimKit version 1.0.45.4 – contain conversion objects.

Disclaimer

EasyIO 30P is a product by EasyIO Corporation Ptd Ltd.

The EasyIO 30P was built on the Sedona Framework ®.

Sedona Framework is a trademark of Tridium, Inc.

Table of Contents

Introduction

This document describe about all the EasyIO Sedona kits and functionality. All the EasyIO Sedona kits can only be used with EasyIO Sedona controllers. No other controllers can be used.

Table below describe about the dependencies for all he EasyIO Sedona kits.

Number	EasyIO Sedona Kit	Current Version	Dependencies	Remarks
1	Easyio	1.0.43.10	Firmware 0.5.00 and later	Must have kit Default kit
2	Easyio30p	1.0.43.0	Easyio 1.0.43.0 or higher	Default kit
3	Easyio30pRegs	1.0.43.0	Easyio 1.0.43.0 or higher	
4	EasyioBacnet	1.0.43.20	Easyio 1.0.43.0 or higher	
5	EasyioComponent	1.0.43.10	Easyio 1.0.43.0 or higher	
6	EasyioControl		Easyio 1.0.43.0 or higher	
7	EasyioDns	1.0.45.2	Easyio 1.0.43.10 Firmware 0.5.00 and later	
8	EasyioEmail	1.0.45.1	Easyio 1.0.43.10 Firmware 0.5.00 and later	
9	EasyioEnergy	1.0.45.1	Easyio 1.0.43.0 or higher EasyIOFGLcd 1.0.45.4 or higher	
10	EasyioFGLcd	1.0.45.1	Easyio 1.0.43.0 or higher	

11	EasyioHistory	1.0.45	Easyio 1.0.43.0 or higher PStore.kit History.kit	
12	EasyioLib	1.0.43.0	Easyio 1.0.43.0 or higher	
13	EasyioLicense	1.0.45	Easyio 1.0.43.10 or higher	
14	EasyioLimkit	1.045.3	Easyio 1.0.43.0 or higher	
15	EasyioMathConversion	1.0.45.21	Easyio 1.0.43.10 or higher Firmware 0.5.00 and later	
16	EasyioModbus	1.0.43.20	Easyio 1.0.43.0 or higher	
17	EasyioModbusSlave	1.0.43.21	Easyio 1.0.43.10 Firmware 0.5.00 and later	
18	EasyNtp	1.0.45	Easyio 1.0.43.10 Firmware 0.5.00 and later	
19	EasyioP2P	1.0.45.1	Easyio 1.0.43.0	
20	EasyioPersistanceControl	1.0.45.22	Easyio 1.0.43.10 Firmware 0.5.00 and later	
21	EasyioSchedule	1.0.45.2	Easyio 1.0.43.0	
22	EasyioSox	1.0.45	Easyio 1.0.43 .0	
23	EasyIOSub	1.0.45	Easyio 1.0.43.0	
24	EasyioTcom	1.0.45	Easyio 1.0.43.0	

25	EasyioTempTable	1.0.45.23	Easyio 1.0.43.10 Firmware 0.5.00 and later	

1. Easyio

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
1	Easyio	1.0.43.10	Firmware 0.5.00 and later	Benchmark EasyIOPlatform

This kit contains 2 objects as show below.

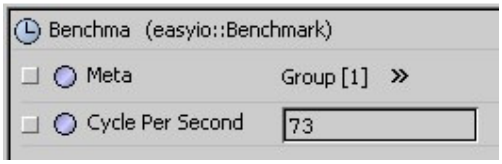
By default an EasyIO Sedona controller comes pre-installed with this kit. To use this object just drag and drop into the wire sheet space.

Benchmark

Benchmark object is an object where is shows the CPU loads with a number. Value varies from 140 to 150 depending on total objects used.
EasyIO component performance measurement, used to measure the single component n cycle per second. The execution time is distributed equally to every installed component, and hence putting one Benchmark component is good enough to benchmark the component execution cycle.

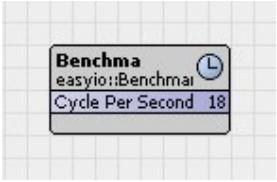
Do not engineer the controller until this *Benchmark* object value drop below "6"

The property sheet of the object is shown below.



- **Cycle Per Second**
  Number of execution cycles per component per second, readonly. This value is not a linear curve versus total number of components.

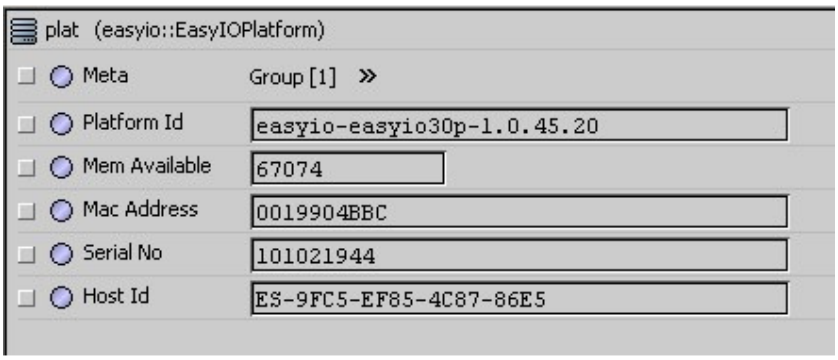
  Note : Do not hit below 6 or else you will be overloading the controller CPU.



Example of the Benchmark object in the wire sheet

1.1. EasyIOPlatform

EasyIOPlatform object is an object where it gets the platform identifier which defines how this Sedona device should be provisioned.
The property sheet of the object is shown below.



- **Platform ID**
  Get the platform identifier which defines how this Sedona device should be provisioned.

- **Mem Available**
  Display available memory of the controller.

- **Mac Address**
  Display the MAC address of the controller.

- **Serial NO**
  Display the serial number of the controller

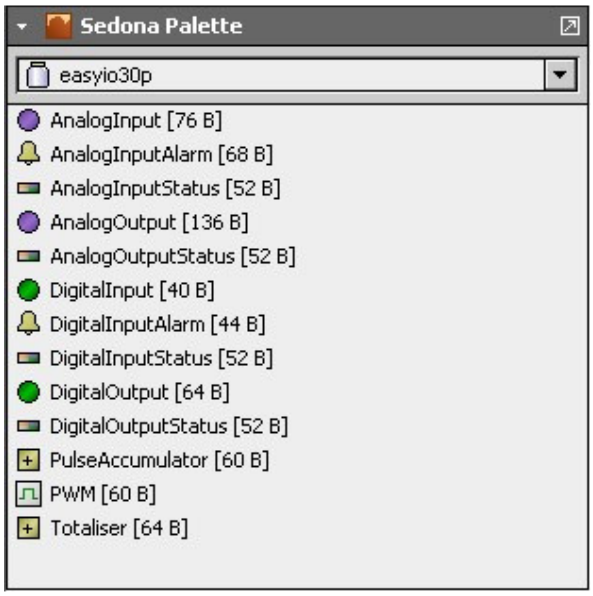
- **Host ID**
  Display the host ID of the controller which will be use for special kit licensing.

2. Easyio30p

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
2	Easyio30p	1.0.43.0	Easyio 1.0.43.00 or higher	AnalogInput AnalogInputAlarm AnalogInputStatus AnalogOutput AnalogOutputStatus DigitalInput DigitalInputAlarm DigitalInputStatus DigitalOutput DigitalOutputStatus PulseAccumulator PWM Totalizer

This kit contains 13 objects. All the objects are to be used for the controller physical I/O points such as Analog Inputs, Analog Outputs, Digital Inputs and Digital Outputs.

To use these objects, simply just drag and drop into the wire sheet.

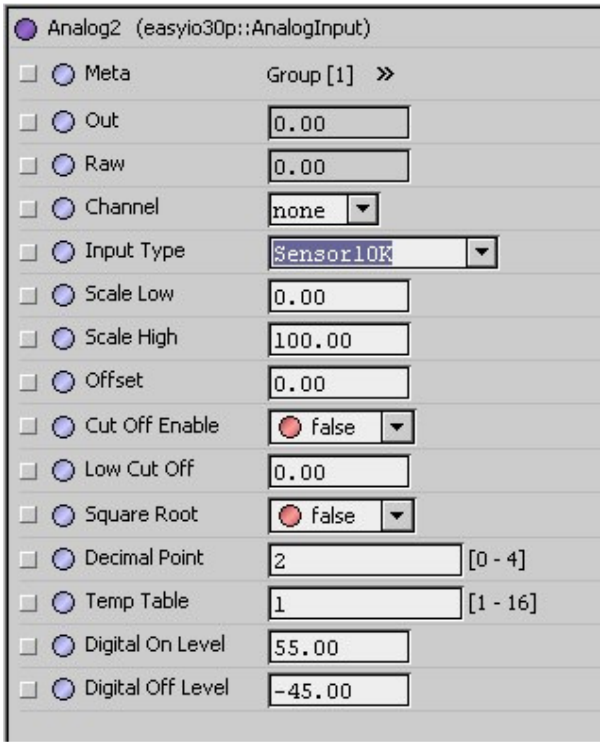


2.1 AnalogInput

AnalogInput component provides a means of reading the analog value connected to one of the physical analog type points on a controller.
There are eight analog input points on EasyIO30P controller that support voltage, current, resistance and temperature sensors. The input type is selected via AI configuration and hardware jumper setting. For temperature sensors, standard curves for 10K Thermistor (with or without 11K shunt), 1K Balco and 1K Platinum (in degree C and Fahrenheit F) are provided within the internal tables.

Additional tables are also available as user defined curves.

The property sheet of the object is shown below.



- **Out**
  The output value of the Analog Input depending of the Input type.

- **Raw**
  The raw value of the Analog Input depending of the input type selection.

- **Channel**
  The input selection channel. Channel 1 – 8

- **Input type**
  Defines the sensor type connected to the physical point and determines the conversion algorithm.

  Current , 4-20mA
  Current, 0-20mA
  Voltage, 0-10V
  Voltage, 0-5V

  Res30K , 30K resistance range
  Res10K, 10K resistance range
  Res1.5k, 1.5K resistance range
  Temperature sensor 30K , 30K range ; 10K thermistor temperature sensor
  Temperature sensor 10K , 10K range ; 10K thermistor sensor with 11k shunt
  Temperature sensor 1.5K , 1.5K range ; 1K platinum temperature sensor

- **Scale Low**

Applicable for voltage and current inputs only.

- **Scale High**
  Applicable for voltage and current inputs only.

- **Offset**
  Offset adjustment for the inputs.

- **Cut off enable**
  Enable the low level cutoff.
  Some sensors might have unstable output at low range operation. The lowCutoff function helps to filter the unstable value by forcing the output value to scale Low value when the input value is lower than the lowCutoff value. The cutoff Enable only applied to current and voltage inputType selection.

- **Low Cut off**
  The cutoff value for AI input value. If the lowCutOff is enabled, the output value will be set to scaleLow value whenever the input value is lower than the lowCutoff Value. The cutoffEnable only applied to current and voltage inputType selection

```
if(AI Value < lowCutOff Value)
 AI Value = scaleLow Value
```

- **Square Root**
  This parameter will only applicable for Input type voltage and current. It will square root the input value.

- **Decimal point**
  Set the roundup decimal point precision of the AI Value during conversion.
  0 – 4

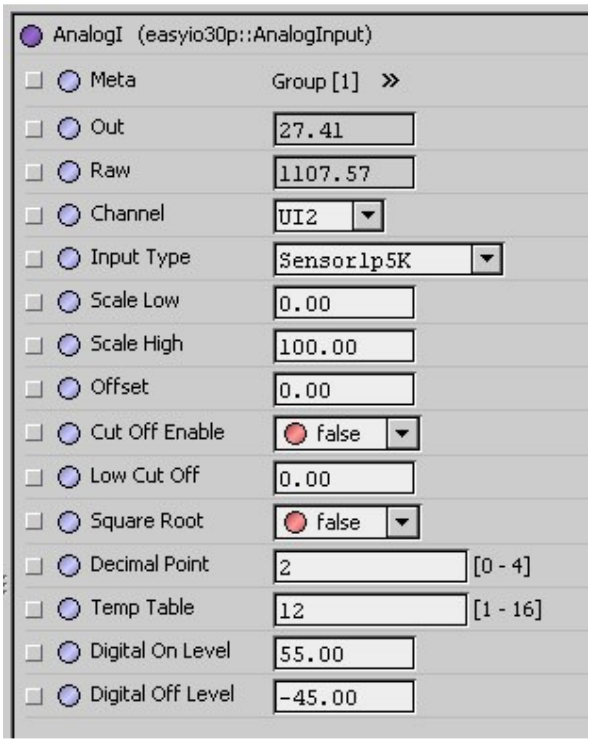
- **Temp table**
  Select analog input temperature table choice.
  1 to 16
  This temperature table defines the temperature curve table index used for lookup conversion for Temperature Sensor InputType selection. The controller has built in 8 default temperature tables (9 to 16) and 8 user defined/customizable temperature tables (1 to 8)

Analog Input Temperature Table Selection

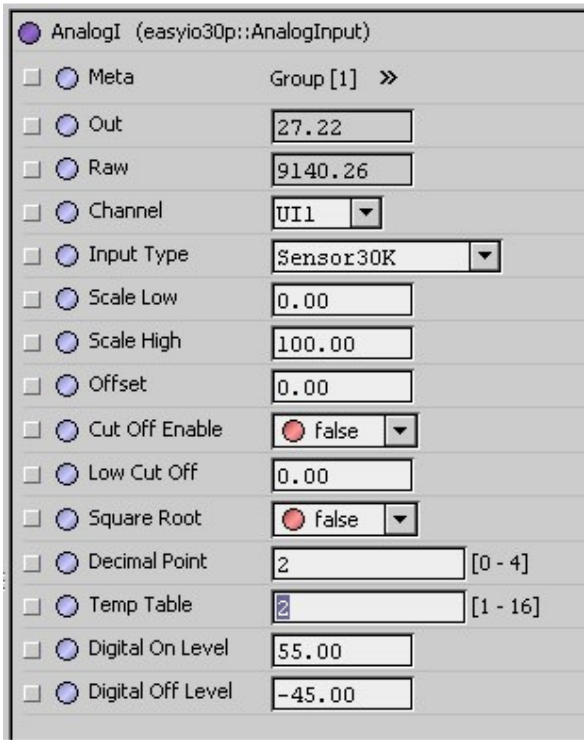
Temperature Table Index	Type of Temperature Sensor
1 – 8	User defined Table 1 – 8 (default = table 9 – 16)
9	10K shunt (11K) Thermistor in Degree C
10	10K Thermistor in Degree C
11	1K Balco in Degree C
12	1K Platinum in Degree C
13	10K shunt (11K) Thermistor in Degree Fahrenheit
14	10K Thermistor in Degree Fahrenheit
15	1K Balco in Degree Fahrenheit
16	1K Platinum in Degree Fahrenheit

- **Digital On level**
  Set the OFF state level of the analog input value for digital transformation. A positive level value means greater than, and a negative level value means lower than during comparison.

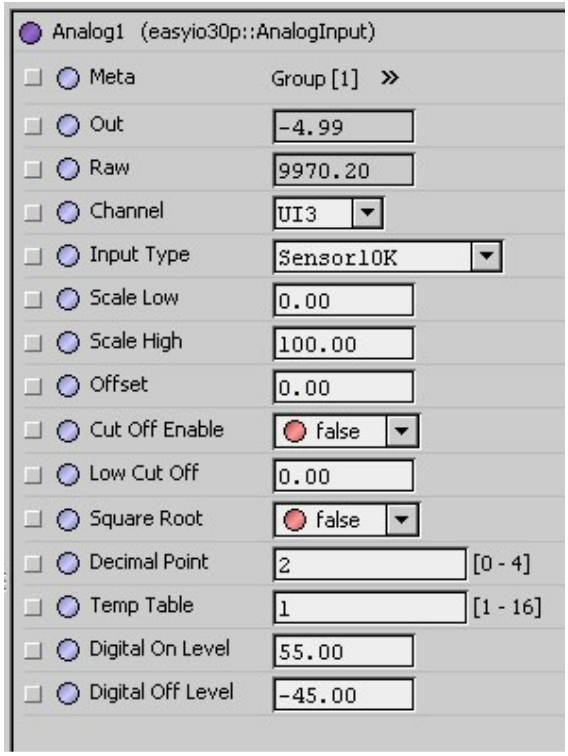
- **Digital Off level**
  Set the OFF state level of the analog input value for digital transformation. A positive level value means greater than, and a negative level value means lower than during comparison.



Example of configuring a 1K Platinum temperature sensor in degree C.



Example of configuring a 10K Thermistor temperature sensor in degree C.Note that selection of Input Type is 30K instead of 10K.

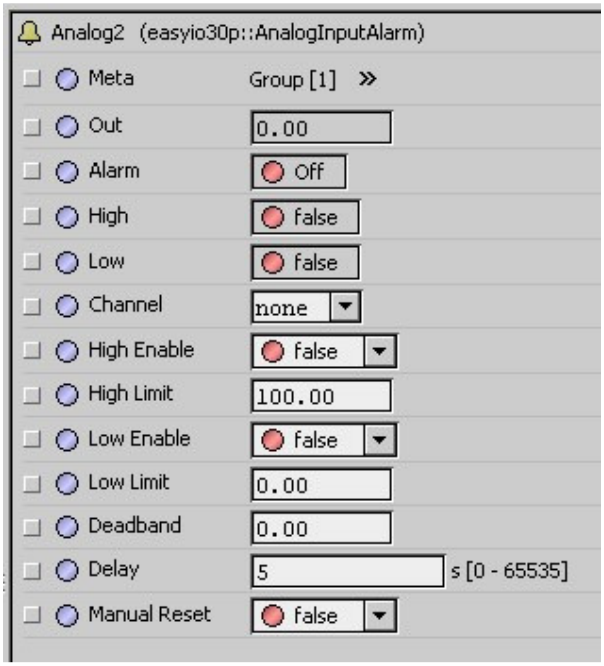


Example of configuring a 10K Thermistor 11K Shunt temperature sensor in degree C. This configuration can be use when configuring Universal Input as Digital Input.

2.2 *AnalogInputAlarm*

AnalogInputAlarm is a EasyIO-30P physical Analog Input (Universal Input) alarm Component. This component is used when user would like to have alarm notification when the Analog value exceed or fall below a pre-define value.

The property sheet of the object is show below.



- **Out**
  The out is the output value of the selected Channel. If the channel is selected as UI1 , then the out is the value of Channel UI1 according to the Input type.

- **Alarm**
  Alarm state of the selected channel base on limit configurations.

- **High**
  Analog alarm state for high limit if high limit is enable. “True = alarm ; False = normal”.

- **Low**
  Analog alarm state for low limit if low limit is enable. “True = alarm ; False = normal”.

- **Channel**
  Channel which this object is tied to. Only applicable for UI1 to UI8.

- **High Enable**
  To enable the high limit

- **High Limit**
  To configure the value for the high limit.

- **Low Enable**
  To configure the value for the high limit.

- **Low Limit**
  To configure the value for the high limit.

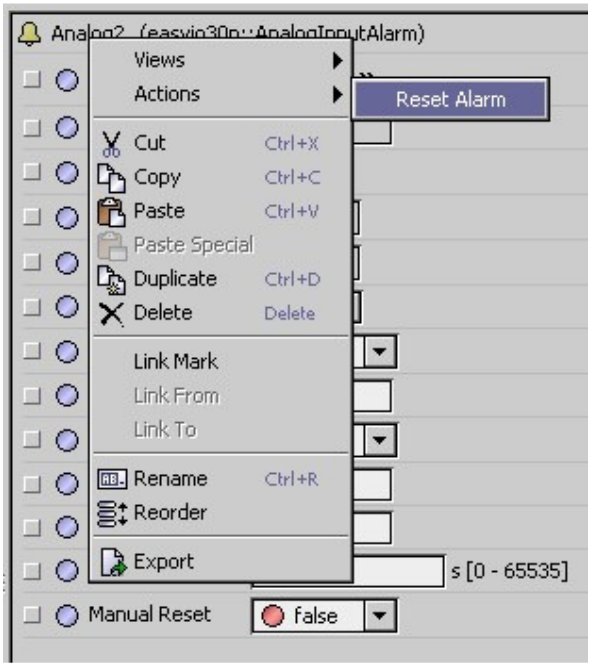
- **Deadband**
  AI alarm deadband value. This deadband is applied to Low and High Alarm Limit values to determine the return from alarm trip points. To return from High Alarm trip point, the AI Value must be lower than the Alarm High limit by Alarm Deadband limit. To return from Low Alarm trip point, the AI Value must be greater than the Alarm Low Limit by Alarm Deadband limit

AI Value	Alarm Condition
> Alarm High Limit	High Alarm
< Alarm High Limit – Alarm Deadband Limit	Return from High Alarm
< Alarm Low Limit	Low Alarm
> Alarm Low Limit + Alarm Deadband Limit	Return from Low Alarm

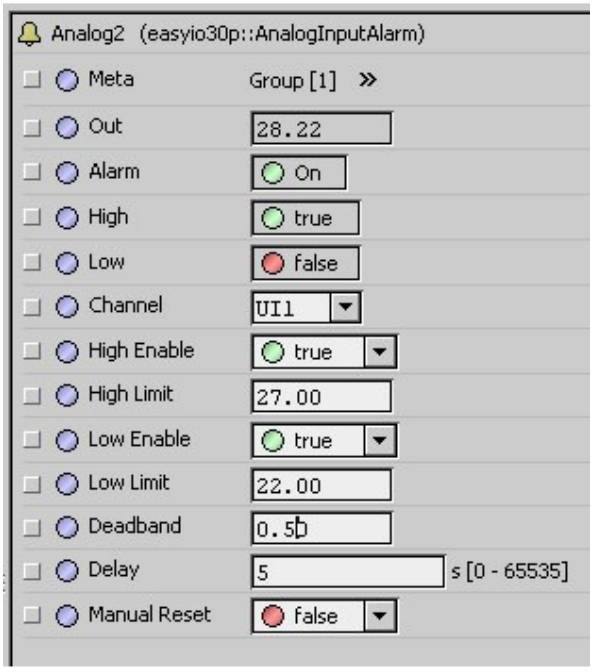
- **Delay**
  AI alarm delay time, maximum 65535 seconds
  Delay time is the duration (in seconds) that the AI Value must be:
  - in the alarm condition before alarm state is generated
  - in the non-alarm condition before returned from alarm state

- **Manual reset**
  Enable/disable AI alarm manual reset.
  Under Auto mode, the Alarm state will be reset when the AI State is in the non-alarm condition. For Manual mode, when alarm is triggered, the Alarm state will stay on even the AI State is back to non-alarm condition.

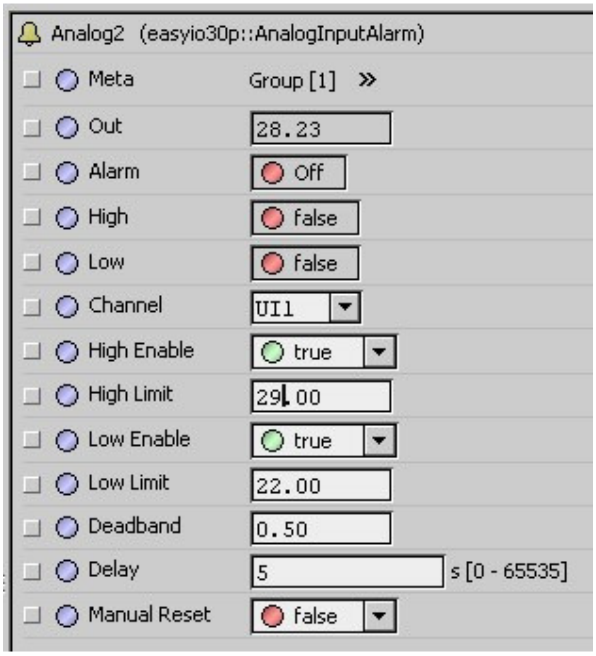
- **ResetAlarm**
  Reset Analog Input alarm state. This only applies to alarm manual reset type. Right click at the object and go to action.



Example of manual reset action.



Example of configuring UI1 fir high limit alarm and low limit alarm.

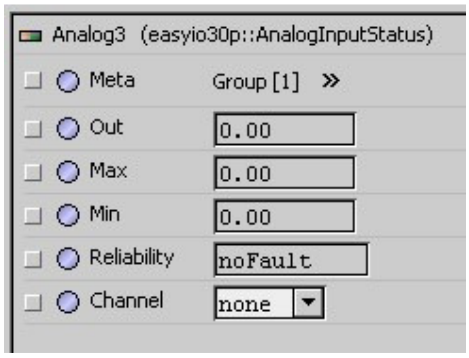


Example of configuring UI1 for high limit alarm and low limit alarm.The alarm has return to normal with manual reset "false"

2.3 *AnalogInputStatus*

AnalogInputStatus is EasyIO-30P physical Analog Input (Universal Input) status Component. It checks the AI configuration and value reliability.

The property sheet of the object is show as below.



- **Out**
  The out is the output value of the selected Channel. If the channel is selected as UI1 , then the out is the value of Channel UI1 according to the Input type.

- **Max**
  This property show the max value of the system since the last system reset or reset action.
- **Min**
  This property show the min value of the system since the last system reset or reset action.

- **Reliability**
  The AI sensor/input conditions when configured as sensor type input.

```
0  = No Fault
1  = Sensor Open
2  = Sensor Short
3  = Over range
4  = Under range
5   = No sensor
```

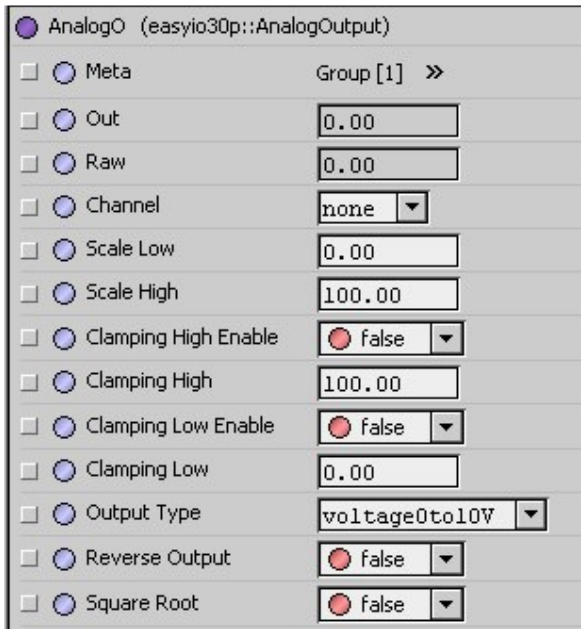
- **Channel**
  Channel which this object is tied to. Only applicable for UI1 to UI8.

2.4 *AnalogOutput*

Analog Output, EasyIO-30P has 4 analog output. The Analogue Output (AO) component provides an interface to the physical analogue output point that can source/drive a 0 to 20mA current or 0 to 10V voltage signal. There are four analogue output points on EasyIO30P controller that support voltage and current. The output type is selected via AO configuration and hardware jumper setting.

The Analogue Output provides 16 levels of prioritized command controls to the analog output. in1 has the highest priority and in16 is the lowest. The inx may have a commanded value (valid floating point value) or a null value (usually NaN in floating point). A null value indicates that there is no value (or not active) at that priority. The analog output continuously monitors all priority inputs (in1 to in16) to locate the entry with the highest priority non-NULL value and sets the output using this value.

The property sheet of the object is show as below.



- **Out**
  The output value of the Analog Output in percentage.

- **Raw**
  The raw value of the Analog Output depending if the Output Type selection. For example; output selection is set to 0V – 10V , the raw value is 0-10

- **Channel**
  The output channel selection. Channel 1 – 4.

- **Scale Low**
  It defines the output value of the AO when the output at the hardware point equals to the lowest value.

```
Voltage (0 - 10V): 0V
Current (0 - 20mA): 0mA
Current (4 - 20mA): 4mA
```

The AO uses the ScaleLow and ScaleHigh to convert the output value to physical value. Raw value register shows the physical output.

```
Physical Value = (Value (%)) * (Scale High - Scale Low)
For instance:

 Type= Current 4 - 20mA
  Scale Low Value = 0%
  Scale High Value = 100%
  Current AO Value = 50%
  Physical AO Value = 12mA (Raw Value)
```

- **Scale High**
  Analog output scale high value.

It defines the output value of the AO when the output at the hardware point equals to the highest value.

```
Voltage (0 – 10V): 10V
Current (0 – 20mA): 20mA
Current (4 – 20mA): 40mA
```

- **Clamping High Enable**
  Enable the clamping high limit control for the Analog Output.

- **Clamping High**
  The high limit of the AO output value. This is to set the max value of the output even if it exceed the process value.

- **Clamping Low Enable**
  Enable the clamping low limit control for the Analog Output.

- **Clamping Low**
  The low limit of the AO output value. This is to set the min value of the output even if it fall below the process value.

- **Output Type**
  Define the output type

```
Voltage , 0-10Volt
Current , 4mA-20mA
Current , 0mA-20mA
```

- **Reverse Output**
  Enable the AO reverse output.
  false = normal output, true = reverse output

  The relationship between reverse output and the output value is as follow:

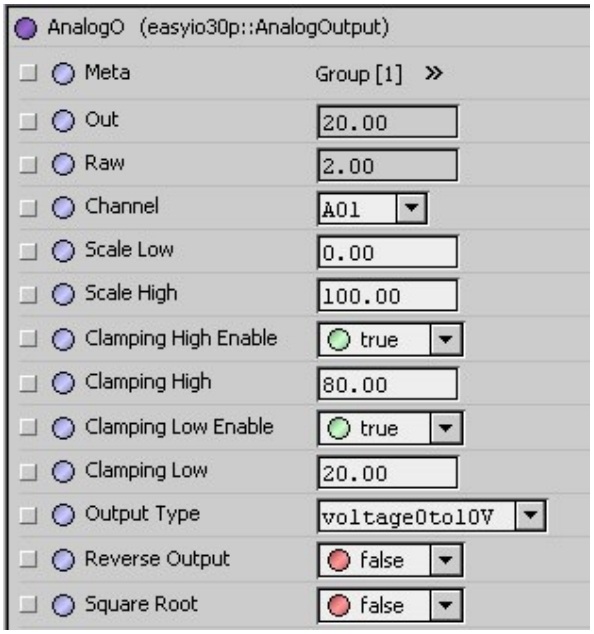
```
Reverse Output (%) = 100 – Output Value (%)
```

- **Square Root**
  Enable the AO square root output.
  false = direct output, true = squareroot output

  The relationship between output value and the physical value (expressed in percentage) is:

```
Physical Calculated Value (%) = Output (%) * Output (%)
Physical Value = Physical Calculated Value (%) * (Output
High – Output Low) + Output Low
```

Where,
Output (%) = 100 * Output Value/ (Scale High Value – Scale Low Value)
Output High = 10V (voltage type), 20mA (current type)
Output Low = 0V (voltage type), 0mA (current type 0 – 20mA), 4mA (current type 4 – 20mA)

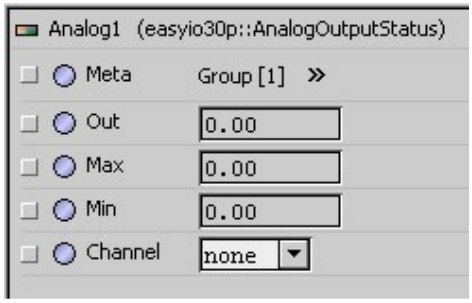


Example of configuring AO1 with output type Voltage 0-10V and clamping high and clamping low enable.

2.5 *AnalogOutputStatus*

Analog Output Status is EasyIO-30P physical Analog output status component. It checks the AO configuration and value reliability.

The property sheet of the object is show as below.



- **Out**
  The out is the output value of the selected Channel. If the channel is selected as AO1 , then the out is the value of Channel AO1 according to the Input type.

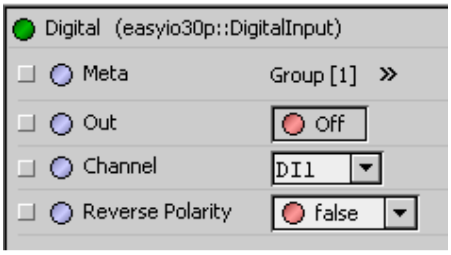
- **Max**
  This property shows the max value of the system since the last system reset or reset action.

- **Min**
  This property shows the min value of the system since the last system reset or reset action.

2.6 *Digital Input*

Digital Input component provides a means of reading the digital value connected to one of the physical input points on the controller. The typical usage is to monitor the status of contact closures from various field devices such as switches, open/close sensors or any other dry contact devices.
There are 16 digital input points on EasyIO30P controller. Eight of them are derived directly from digital input detection circuitry (+5Vdc pulled up), named DI1 to DI8 whereby the other eight inputs are derived from universal input (UI or AI) using value conversion (D9 to D16). DI1 to DI8 might have different characteristics if compare to DI9 to DI16 depend on the AI settings. Please refer to AI component section for the conversion.

The property sheet of the object is show as below.



- **Out**
  This value is the current digital input state
  False = Off , True = On

- **Channel**
  This parameter defines the digital input channel.

  None = No input selected,
  DI1-DI8 = digital input,
  DI9 -DI16 = digital input derived from Universal Inputs.

- **Reverse Polarity**
  Reverse polarity reverse controls the relationship between the physical digital input and the digital input state.

  If reversePolarity is false, the output value (out) to directly reflect the digital condition of the physical point. An active state (closed contact) is considered ON while inactive state (open contact) is considered OFF.

  If reversePolarity is true, the output value (out) to inversely reflect the digital condition of the physical point. An active state (closed contact) is considered OFF while inactive state (open contact) is considered ON.

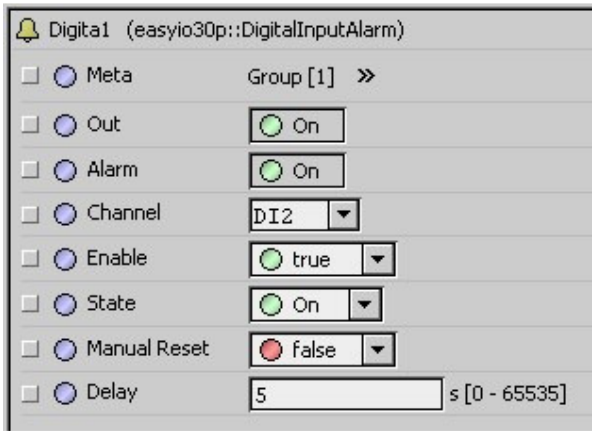
Digital Input Polarity

Physical Hardware	Physical State	Polarity	DI State
Open Contact	Active	Direct	ON
Closed Contact	Inactive	Direct	OFF
Open Contact	Active	Reverse	OFF
Closed Contact	Inactive	Reverse	ON

2.7 *Digital Input Alarm*

Digital Input Alarm is a EasyIO-30P physical Digital Input alarm Component. This component is used when user would like to have alarm notification when the Digital Input in true state.

The property sheet of the object is show as below.



- **Out**
  Out is the selected digital channel input state.

- **Alarm**
  Alarm state of the selected channel.

- **Channel**
  None = No input selected,
  DI1-DI8 = digital input,
  DI9 -DI16 = digital input derived from Universal Inputs.

- **Enable**
  This parameter is to enable the digital input alarm.

- **State**
  Digital input alarm monitoring state.

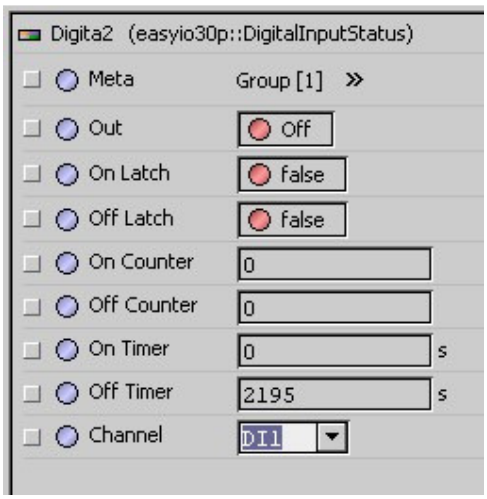
- **Manual reset**
  Enable/disable AI alarm manual reset.
  Under Auto mode, the Alarm state will be reset when the DI State is in the non-alarm condition. For Manual mode, when alarm is triggered, the Alarm state will stay on even the DI State is back to non-alarm condition

- **Delay**
  AI alarm delay time, maximum 65535 seconds
  Delay time is the duration (in seconds) that the AI Value must be:
  - in the alarm condition before alarm state is generated
  - in the non-alarm condition before returned from alarm state

2.8 *Digital Input Status*

Digital Input Status is EasyIO-30P physical Digital Input status Component. It checks the DI state and as well time in particular state.

The property sheet of the object is show as below.



- **Out**
  The out is the output value of the selected Channel. If the channel is selected as DI1.

- **On Latch**
  The ON to OFF transition captured at the digital input state. Read-only

- **Off Latch**
  The ON to OFF transition captured at the digital input state. Read only

- **On Counter**
  Digital input state OFF to ON transition counter. Read only
  This property increments by one on each digital input state changed from OFF to ON. Read-only

- **Off Counter**
  Digital input state ON to OFF transition counter. Read only
  This property increments by one on each digital input state changed from OFF to ON. Read-only

- **On Time**
  The duration (in seconds) of the digital input state remains in the ON state. Read-only
  The On Timer is reset to 0 automatically when an OFF to ON transition is occurred at the digital input state and it will hold the value when the digital input state is at OFF state. Read-only

- **Off Timer**
  The duration (in seconds) of the digital input state remains in the OFF state. Read-only
  The Off Timer is reset to 0 automatically when an OFF to ON transition is occurred at the digital input state and it will hold the value when the digital input state is at OFF state. Read-only

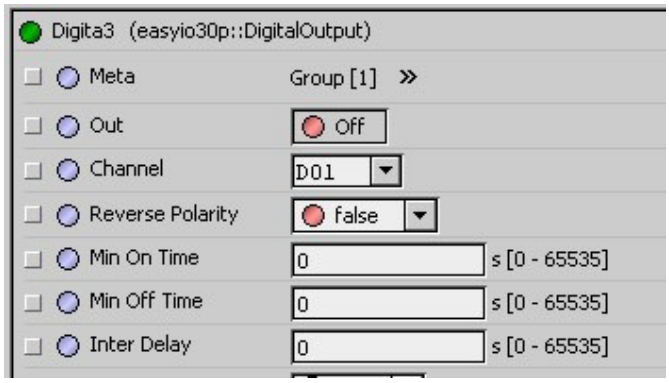
- **Channel**
  None = No input selected,
  DI1-DI8 = digital input,
  DI9 -DI16 = digital input derived from Universal Inputs.

2.9 *Digital Output*

Digital Output EasyIO-30P has 8 digital output control. The Digital Output component provides a means of turning a physical digital output point OFF or ON. The typical usage is for start/stop controls of external equipment such light, valve, fan or any other digital control equipment. The DO component monitors the required set state and determines the proper hardware output action based on its settings. There are eight digital output points on EasyIO30P controller. Each of them is driven by a dry contact relay (SPST Relay) which is able to drive the external devices up to 1 Ampere (AC/DC).

Digital output is a prioritized command with 16 priorities control plus a default value (relinquish default). in1 has the highest priority & in16 has the lowest priority. in6 is reserved for minimum/maximum on time control. The value can be commanded value (false = 0, true = 1) or a null value (= 2). A null value indicates that there is no value (or not active) at that priority.

The property sheet of the object is show as below.



- **Out**
  The current Digital Output State.

- **Channel**
  The output channel selection. Channel DO1 – DO8

- **Reverse Polarity** controls the relationship between the physical digital command state.

  If reversePolarity is false, the out state to directly reflect the digital condition of the command state. An active state (closed contact) is considered ON while inactive state (open contact) is considered OFF.

  If reversePolarity is true, the out state to inversely reflect the digital condition of the command state. An active state (closed contact) is considered OFF while inactive state (open contact) is considered ON.

Digital Output Polarity

Command State	Polarity	Physical State	Output State
OFF	Direct	Inactive	Open Contact
ON	Direct	Active	Closed Contact
OFF	Reverse	Active	Closed Contact
ON	Reverse	Inactive	Open Contact

- **Min On Time**
  The minOnTime prevents the out state from being changed to OFF state from ON state for a specified time. This ensures that the out state will stay ON for a minimum period (in seconds) before it can be turned off. This prevents short-cycling and helps to increase equipment life-cycle. Maximum = 65535

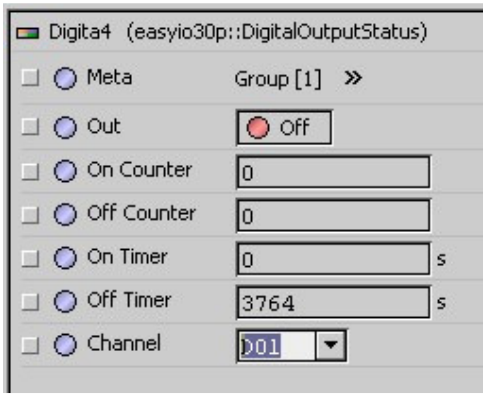
- **Min Off Time**
  The minOffTime prevents the out state from being changed to ON state from OFF state for a specified time. This ensures that the out state will stay OFF for a minimum period (in seconds) before it can be turned off. This prevents short-cycling and help to increase equipment life-cycle. Maximum = 65535

- **Inter Delay**
  The interDelay prevents other digital output objects to change theirs state for a specific time (in seconds) after its state has changed. This protection prevents equipments from turning on and off at the same time, hence reduces overloading, high spike surge and other electrical problems. Maximum = 65535

2.10 *Digital Output Status*

Digital Output Status is EasyIO-30P physical Digital Input status Component. It checks the DI state and as well time in particular state.

The property sheet of the object is show as below.



- **Out**
  The current Digital Output State according to selected Output Channel

- **On Counter**
  Digital input state OFF to ON transition counter. This property increments by one on each digital output state changed from OFF to ON. Readonly

- **Off Counter**
  Digital input state ON to OFF transition counter. This property increments by one on each digital output state changed from ON to OFF. Readonly

- **On Timer**
  The duration (in seconds) of the digital output state remains in the ON state. The onTimer is reset to 0 automatically when an OFF to ON transition is occurred at the digital output state and it will hold the value when the digital output state is at OFF state. Readonly

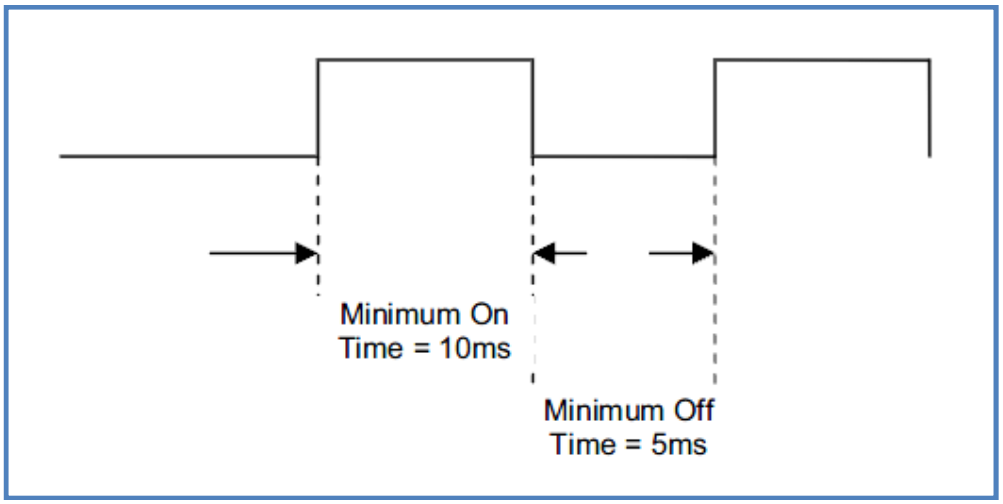
- **Off Timer**
  The duration (in seconds) of the digital output state remains in the OFF state. The offTimer is reset to 0 automatically when an ON to OFF transition is occurred at the digital output state and it will hold the value when the digital output state is at ON state. Readonly

- **Channel**
  The output channel selection. Channel DO1 – DO8

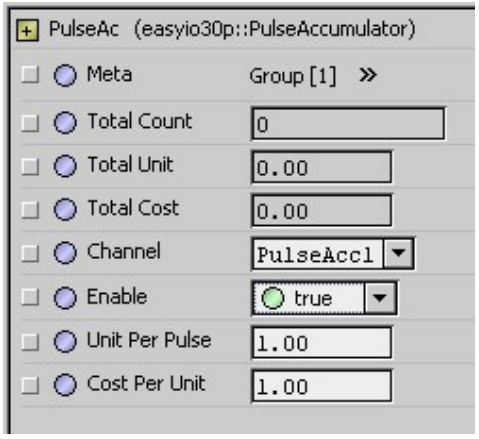
2.11 *Pulse Accumulator*

Pulse Accumulator has 4 inputs. The Pulse Accumulator component detects and accumulating the digital pulse input and logging it to the non-volatile memory. The totalCount data will be written to non-volatile memory every 60 seconds. Only DI1, DI2, DI3 and DI4 are able to take the digital pulse input. The minimum on duration of the pulse width is 10ms and off duration is 5ms.

Image below show the minimum pulse width for ON and OFF



The property sheet of the object is show as below.



- **Total Count**
  The accumulated counts of the digital pulse input. This data will be logged into non-volatile memory every 60 seconds

- **Total Unit**
  Pulse Accumulator current total unit

  The max pulse count is 4,294,967,295 (4.2 billion pulse count)

- **Total Cost**
  Pulse Accumulator current total cost

- **Channel**
  Pulse Accumulator channel

  Pulse Accumulator 1 = Digital Input 1
  Pulse Accumulator 2 = Digital Input 2
  Pulse Accumulator 3 = Digital Input 3
  Pulse Accumulator 4 = Digital Input 4

- **Enable**
  Enable Pulse Accumulator

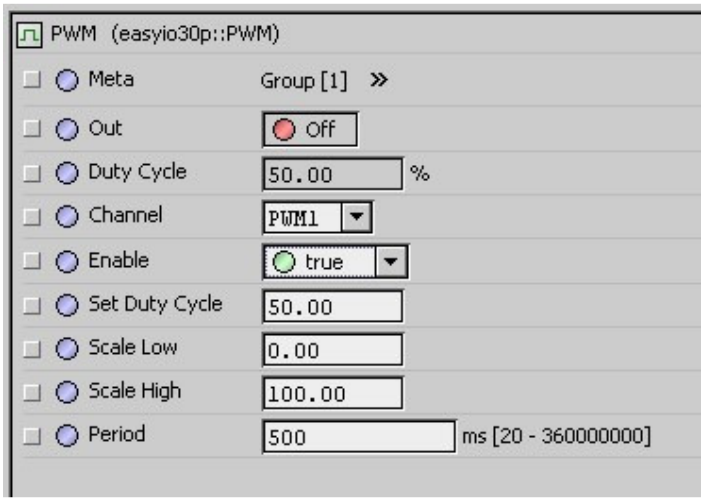
- **Unit per Pulse**
  The number of units represent by a pulse input.

- **Cost per Unit**
  The number of cost per unit.

2.12 *PWM*

PWM (The Pulse Width Modulation) component provides a time proportioned On/Off digital output signal in response to a 0 to 100% input signal (duty cycle, either fixed or dynamic). The digital On/Off cycle operation is determined by the time period and the duty cycle. This output can be applied for both fixed and compensated duty cycle applications like control valves, actuators, electric heat loads and etc. There are two PWM output points on EasyIO30P controller. The PWM outputs are driven by isolated open collector transistor (3.75KV isolated) which able to sink 1A current at maximum 60Vdc.

The property sheet of the object is show as below.



- **Out**
  The PWM output state

- **Duty Cycle**
  PWM current duty cycle.
  The DutyCycle shows the working duty cycle for the PWM output based on the SetDutyCycle, ScaleLow and ScaleHigh properties.

- **Channel**
  The PWM channel.
  0 = No PWM selected,
  PWM 1 = OC 1
  PWM 2 = OC2

- **Enable**
  Enable PWM
  When PWM output control is disabled, the physical output will be set off state at all time.

- **Set Duty Cycle**

PWM set duty cycle

The DutyCycle input determines the on and off duration of the PWM output for one Period time. The input value will be scaled using ScaleLow and ScaleHigh value.

$$\text{Working Duty Cycle \%} = \frac{\text{Set Duty Cycle} - \text{Duty Cycle Low Scale}}{(\text{Duty Cycle High Scale} - \text{Duty Cycle Low Scale})} \times 100\%$$

If the SetDutyCycle is lower than the ScaleLow value, then the Working Duty Cycle will be set to 0%, and if the SetDutyCycle is higher than the ScaleHigh value, then the Working Duty Cycle will be set to 100%. A 0% working duty cycle will set the PWM output to off state and 100% working duty cycle will be set the PWM output to on state during the cycle time. Table below shows how the output on and off time will be affected by the working duty cycle.

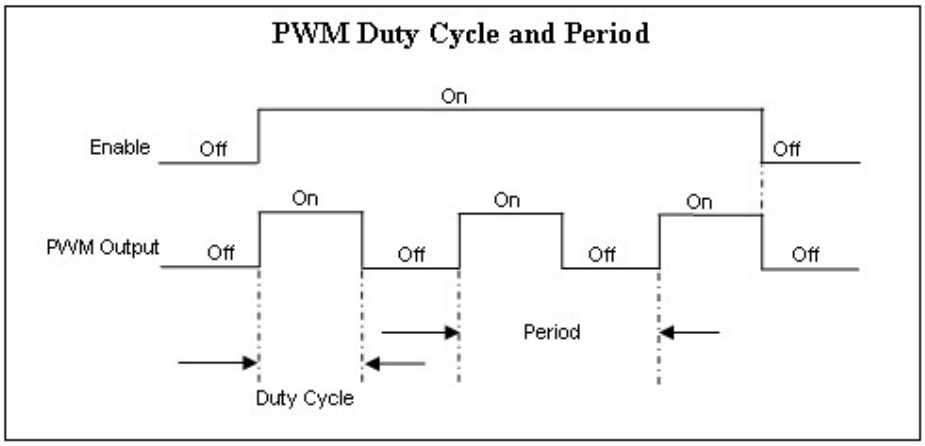
PWM Duty Cycle and On/Off Time

Working Duty Cycle	On Time	Off Time
0.00	Off all the time	
10.00	0.10 x Period	0.90 x Period
25.00	0.25 x Period	0.75 x Period
40.00	0.40 x Period	0.60 x Period
50.00	0.50 x Period	0.50 x Period
75.00	0.75 x Period	0.25 x Period
90.00	0.90 x Period	0.10 x Period
100.00	On all the time	

- **Scale Low**
  Defines the Duty Cycle lowest value which is equivalent to 0%.

- **Scale High**
  Defines the Duty Cycle highest value which is equivalent to 100%.

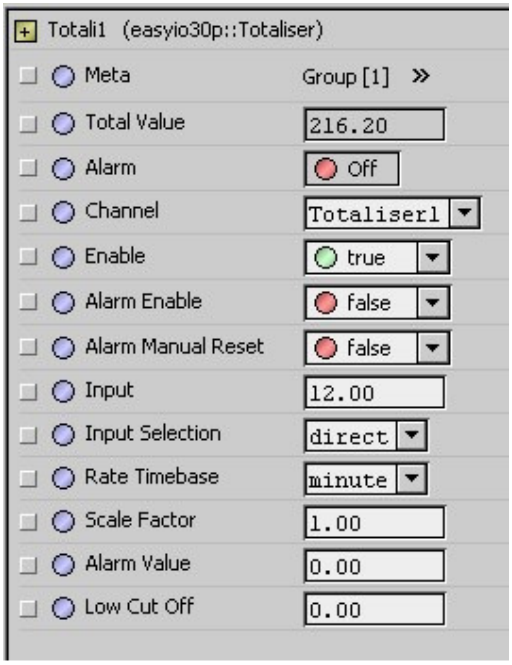
- **Period**
  Defines the PWM period in milli-second
  The Period input defines the repeating time for one complete On/Off cycle for the PWM output. The Period may range from 20ms to 360000000ms (100 Hours).



2.13 *Totalizer*

Totalizer component provides an accumulator function for analog input. Normally, the analog input is the flow rate measurement value. The accumulated value is stored in the non-volatile memory. It also provides alarm monitoring for the accumulated value.

The property sheet of the object is show as below.



- **Total Value**
  Totalizer current accumulated value. Readonly

- **Alarm**
  Totalizer current alarm state. Alarm will only occur if Alarm Enable = true.

- **Channel**
  Total number of 8 object are available for use.

- **Enable**
  Totalizer enable

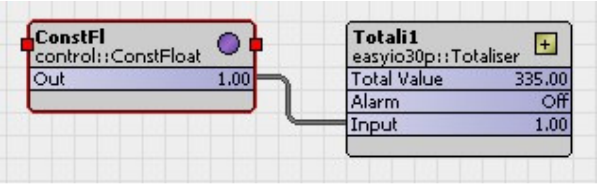
- **Alarm Enable**
  Totalizer alarm monitoring enable.

- **Alarm Manual Reset**
  To set alarm reset type.
  False = auto reset
  True = manual reset is required

- **Input**
  Totalizer input value

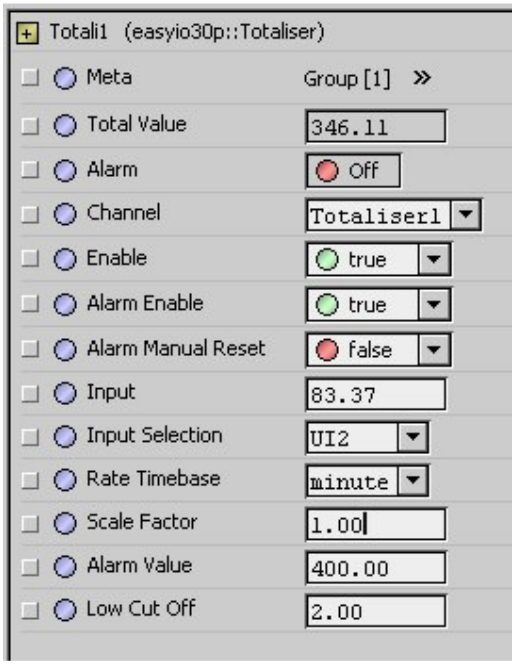
- **Input Selection**
  Define totalizer input source.

  Direct = Input that been link to the object



Example of direct input to the totalizer object

  UI1 – UI8 = Input source from UI channel.



Example of universal input to the totalizer object. The input is from a UI2 value.

- **Rate Timebase**
  Set Totalizer time base used for accumulation

  ```
  Second
  Minute
  Hour
  ```

- **Scale Factor**
  Set Totalizer scale factor for accumulated value
  The ScaleFactor provides scale up or scale down function for the accumulated value. The scale factor can be changed at anytime. For instance, to change liter/minute to milliliter/minute, set the scale factor to 1000.

- **Alarm Value**
  Totaliser accumulated value for alarm activation. The TotalValue is monitored and compared to this value to initiate the Alarm sequence.

- **Low Cut Off**
  Set Totalizer cutoff value
  Some sensor might have unstable output at low range operation. The LowCutoff function helps to filter the unstable value by forcing the output value to 0 when the input value is lower than the LowCutoff value.

3 Easyio30pRegs

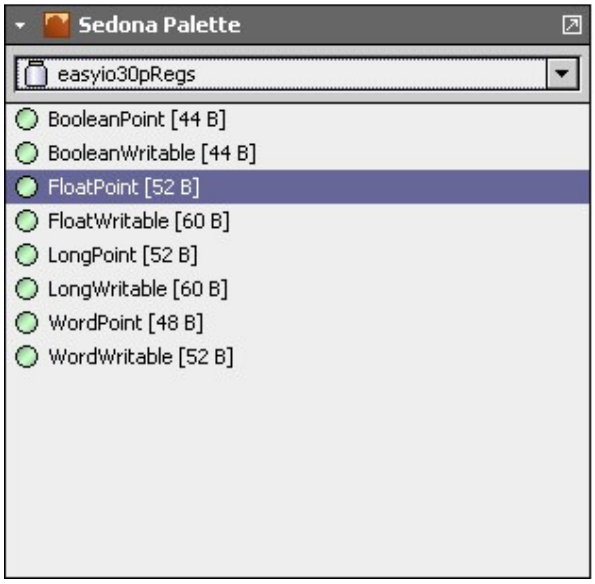
Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
3	Easyio30pRegs	1.0.43.00	Easyio 1.0.43.00 or higher	BooleanPoint BooleanWritable FloatPoint FloatWritable LongPoint LongWritable WordPoint WordWritable

This kit contains 8 objects. All the objects are to be use to access registers from the controller other than the physical I/O points.

Some example of register that can be use in the controller are ;

- Modbus Comm Monitoring
- Bacnet Comm Monitoring

To use these objects just drag and drop into the wire sheet.



3.1 *Boolean Point*

BooleanPoint is use to access to EasyIO-30P boolean point type (readonly) register = Modbus Discrete Input. The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

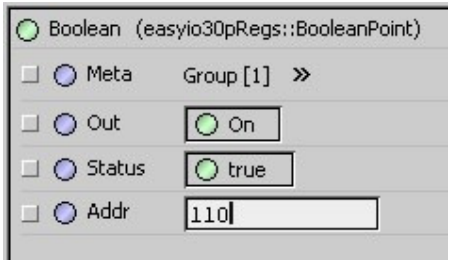
The property sheet of the object is shown below.

- **Out**
  Boolean Point registers current value. Readonly

- **Status**
  Boolean Point registers validity. Readonly
  True = valid
  False = invalid

- **Address**
  Boolean Point register address

Example showing Boolean point with the register D:111



Example showing Boolean register access from the EasyIO , note that it is based-1. Which means the web register has to minus 1 in the sedona.

3.2 *Boolean Writable*

BooleanWritable is use to access to EasyIO-30P boolean writable type register = Modbus Coil Output. The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

The property sheet of the object is shown below.

- **Out**
  Boolean Writable registers current value.

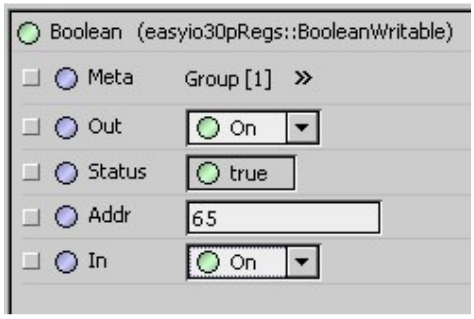
- **Status**
  Boolean Writable registers validity.
  True = valid
  False = invalid

- **Address**
  Boolean Writable register address

- **In**
  Local input value.

Parameters	
Digital Input Polarity 1 (C:66)	○ Direct ◉ Reverse
Digital Input Alarm Enable 1 (C:82)	◉ Disabled ○ Enabled

Example showing Boolean point with the register C:66

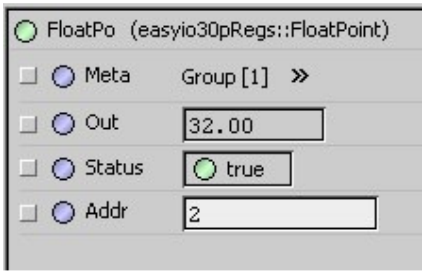


Example showing Boolean Writable register access from the EasyIO , note that it is based-1. Which means the web register has to minus 1 in the sedona.

3.3 *Float Point*

FloatPoint is use to access to EasyIO-30P Float Point type (readonly) register = Modbus Input register (floating point). The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

The property sheet of the object is shown below.



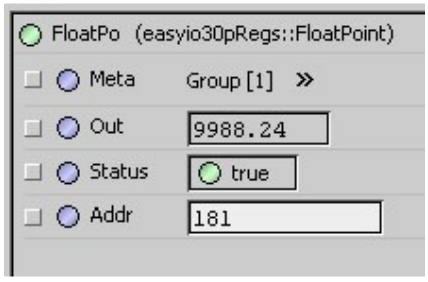
- **Out**
  Float Point registers current value. Readonly

- **Status**
  Float Point registers validity. Readonly
  True = valid
  False = invalid

- **Address**
  Float Point register address

Analogue Input Value 1 (I:1)	-4.9900	
Analogue Input Raw Value 1 (I:182)	9988.2363	mA/V/Ohm

Example showing Float point with the register I:182

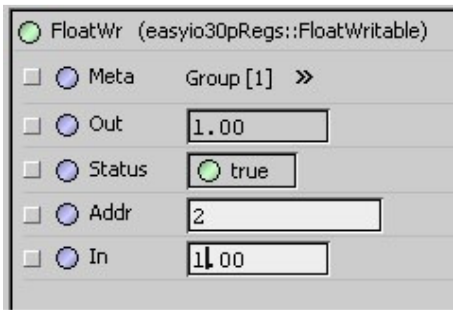


Example showing Float Point register access from the EasyIO , note that it is based-1. Which means the web register has to minus 1 in the sedona.

3.4 *FloatWritable*

FloatWritable is use to access to EasyIO-30P Float Writable type register = Modbus Holding register (floating point). The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

The property sheet of the object is shown below.



- **Out**
  Float Writable registers current value.

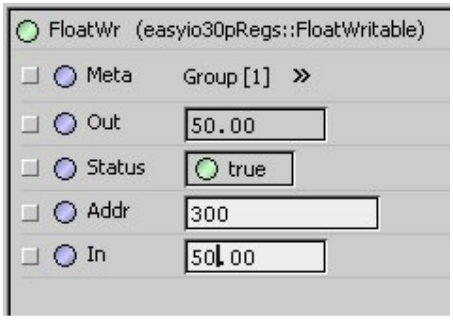
- **Status**
  Float Writable registers validity.
  True = valid
  False = invalid

- **Address**
  Float Writable register address

- **In**
  Local input value.

Analogue Output Scale Low 1 (H:301)	0.0000
Analogue Output Scale High 1 (H:309)	100.0000

Example showing Float Writable with the register H:301

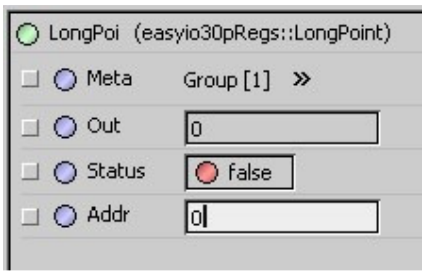


Example showing Float Writable register access from the EasyIO , note that it is based-1. Which means the web register has to minus 1 in the sedona.

3.5 *LongPoint*

LongPoint is use to access to EasyIO-30P Long Point type (readonly) register = Modbus Input register (32-bit integer). The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

The property sheet of the object is shown below.



- **Out**
  Long Point registers current value. Readonly

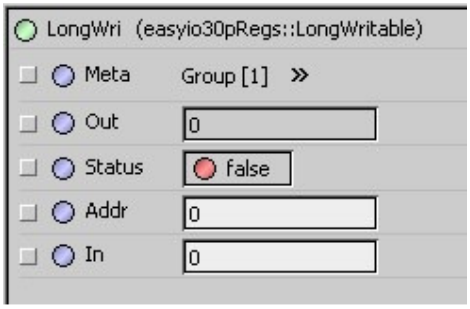
- **Status**
  Long Point registers validity. Readonly
  True = valid
  False = invalid

- **Address**
  Long Point register address

3.6 *LongWritable*

LongWritable is use to access to EasyIO-30P Long Writable type register = Modbus Holding register (32-bit integer). The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

The property sheet of the object is shown below.



- **Out**
  LongWritable registers current value.

- **Status**
  Long Writable registers validity.
  True = valid
  False = invalid

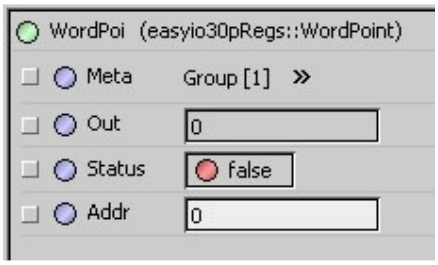
- **Address**
  Long Writable register address

- **In**
  Local input value.

3.7 *WordPoint*

WordPoint is used to access to EasyIO-30P Word Point type (readonly) register = Modbus Input register (16-bit integer). The register address is based-0 address whereby EasyIO-30P documented address is based-1 address

The property sheet of the object is shown below



- **Out**

Word Point or Integer Point registers current value. Readonly

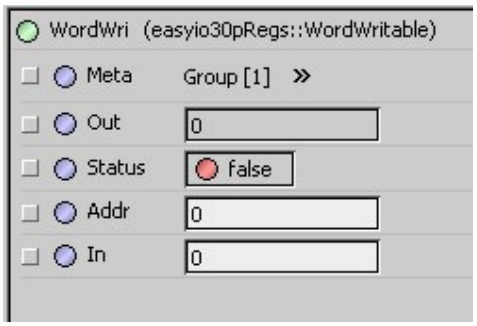
- **Status**
  Word Point or Integer Point registers validity. Readonly
  True = valid
  False = invalid

- **Address**
  Word Point or Integer Point register address

3.8 *WordWritable*

WordWritable is use to access to EasyIO-30P Word Writable type register = Modbus Holding register (16-bit integer). The register address is based-0 address whereby EasyIO-30P documented address is based-1 address.

The property sheet of the object is shown below



- **Out**
  LongWritable registers current value.

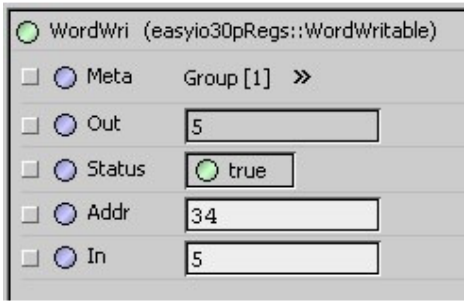
- **Status**
  Long Writable registers validity.
  True = valid
  False = invalid

- **Address**
  Long Writable register address

- **In**
  Local input value.

Modbus TCP/IP Port (H:34)	502
Modbus Serial Turn Around Time (H:35)	5
Modbus Bridge/Slave Response Time Out (H:36)	500

Example showing Word Writable with the register H:35



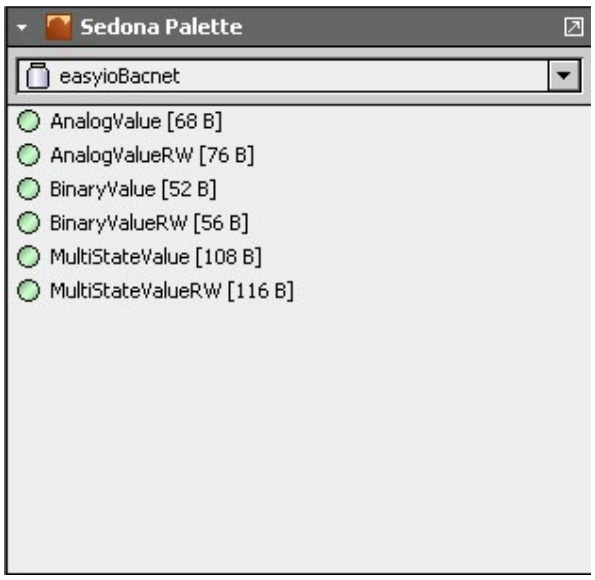
Example showing Word Writable register access from the EasyIO , note that it is based-1. Which means the web register has to minus 1 in the sedona.

4 EasyioBacnet

Number	EasyIO Sedona Kit	Current Version	Dependencies	Remarks
4	EasyioBacnet	1.0.43.20	Easyio 1.0.43.00 or higher	AnalogValue AnalogValueRW BinaryValue BinaryValueRW MultiStateValue MultiStateRW

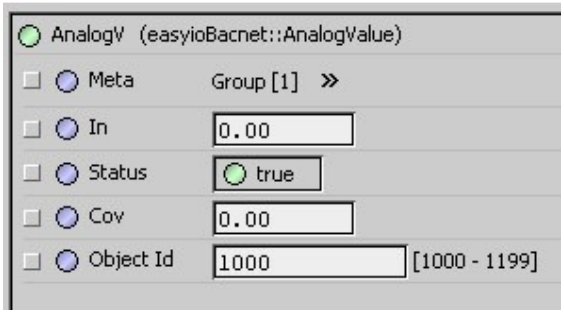
This kit contains 6 objects. All the objects are to be used for BACnet points broadcast. Each object has a max of 200 register.

To use these objects just drag and drop into the wire sheet.



4.1 *AnalogValue*

AnalogValue , Maximum 200 Analog Value objects can be defined (ID: 1000 - 1199).
The property sheet of the object is shown below

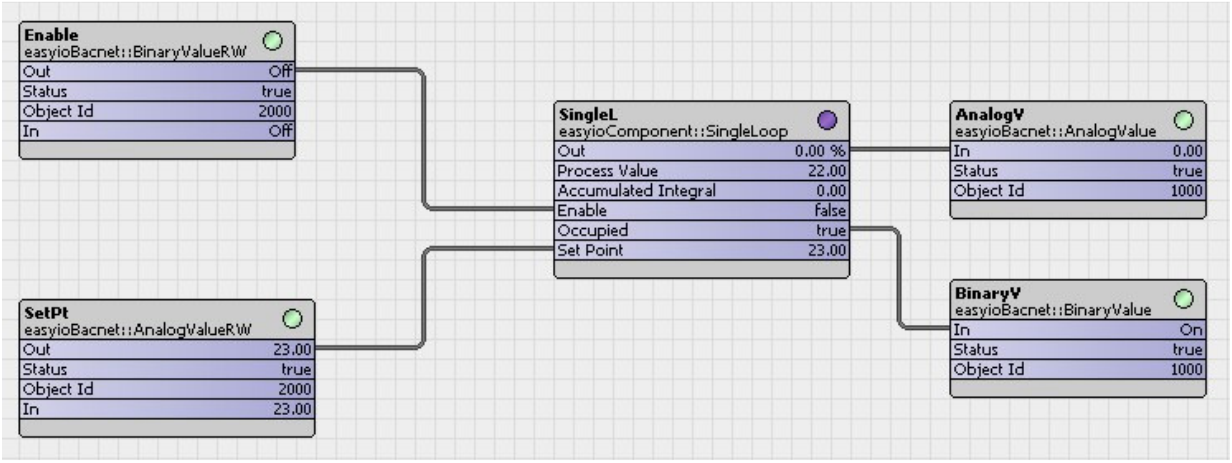


- **In**
  Bacnet Analog current input value. This is normally link from an analog input.

- **Status**
  Bacnet Analog Value object status. Readonly
  True = valid
  Fasle = invalid

- **Cov**
  Bacnet Analog Value cov

- **Object Id**
  Bacnet Analog Value object ID
  1000 – 1199

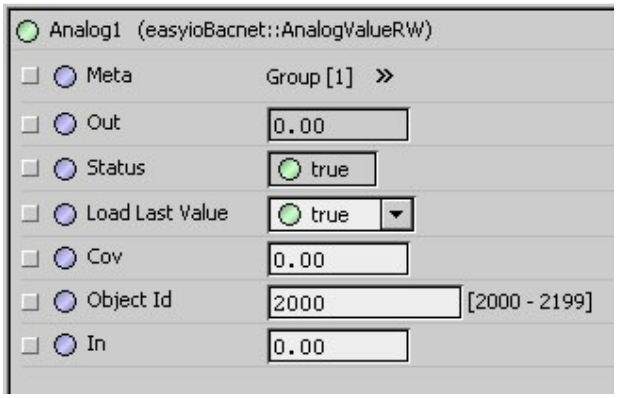


Example of using the AnalogValue , reading the PID loop output. Noticed the bacnet object ID = 1000

4.2 *AnalogValueRW*

AnalogValueRW Maximum 200 Analog Value RW objects can be defined (ID: 2000 - 2199).

The property sheet of the object is shown below



- **Out**
  Bacnet Analog Value current output value.

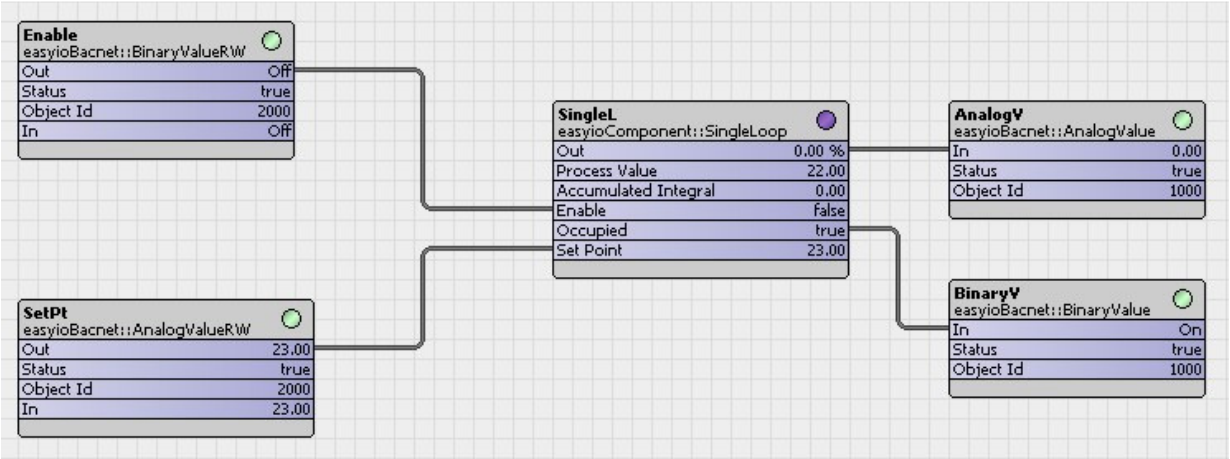
- **Status**
  Bacnet Analog Value object status. Readonly
  true = valid
  Fasle = invalid

- **Load Last Value**
  Enable the load last value when program start
  true = load last stored value when start
  false = do not load last value

- **Cov**
  Bacnet Analog Value cov. This is to set the COV increment.

- **Object Id**
  Bacnet Analog Value object ID
  1000 – 1199

- **In**
  Bacnet Analog Value object input value.

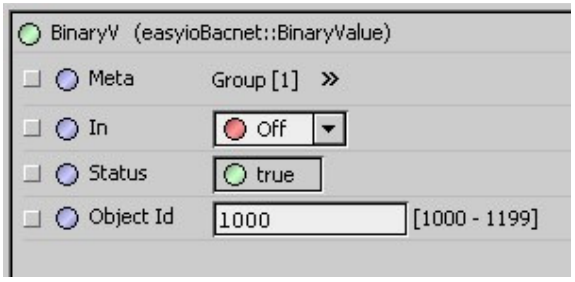


Example of using the AnalogValueRW ,writing the PID loop setpoint. Noticed the bacnet object ID = 2000

4.3 *BinaryValue*

BinaryValue maximum 200 Binary Value objects can be defined (ID: 2000 - 2199).

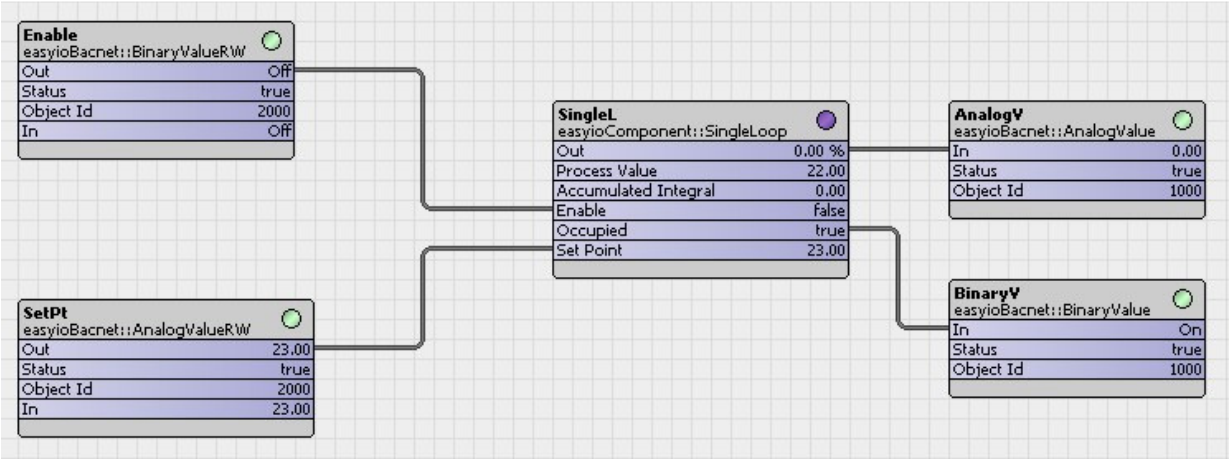
The property sheet of the object is shown below



- **In**
  Bacnet Binary current input value. This is normally link from an analog input.

- **Status**
  Bacnet Binary Value object status. Readonly
  true = valid
  Fasle = invalid

- **Object Id**
  Bacnet Binary Value object ID
  1000 – 1199

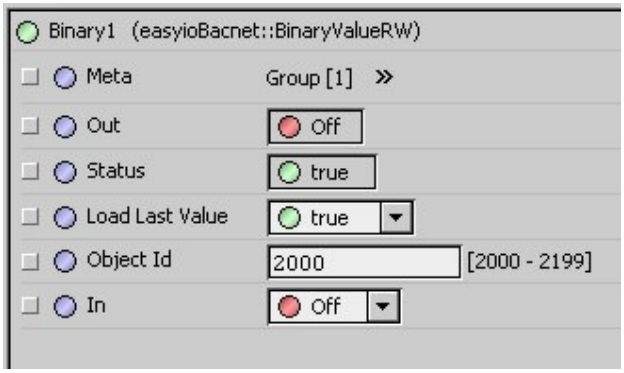


Example of using the BinaryValue , reading the PID loop occupancy status. Noticed the bacnet object ID = 1000

4.4 *BinaryValueRW*

BinaryValueRW maximum 200 Binary Value RW objects can be defined (ID: 2000 - 2199).

The property sheet of the object is shown below



- **Out**
  Bacnet Binary Value current output value.

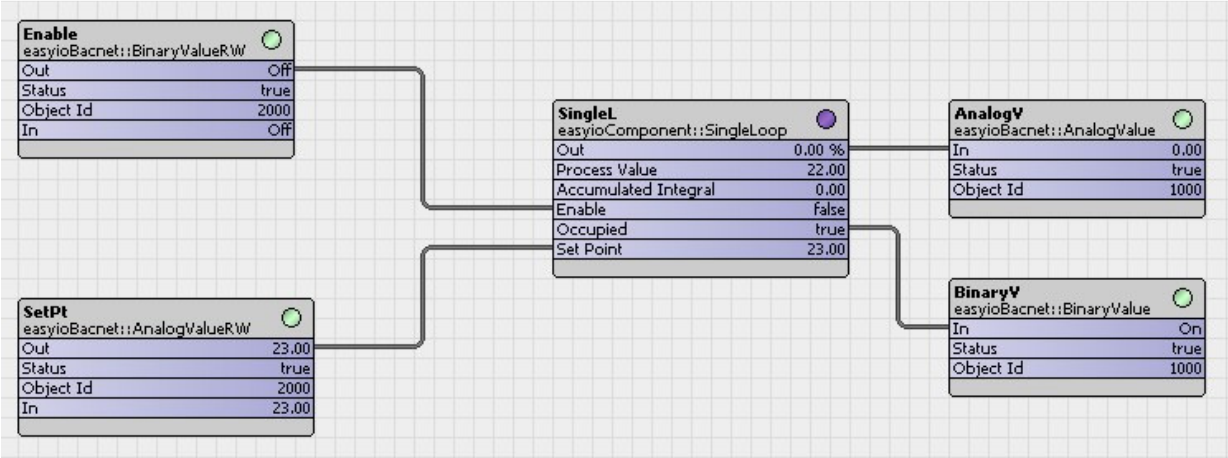
- **Status**
  Bacnet Binary Value object status. Readonly
  true = valid
  Fasle = invalid

- **Load Last Value**
  Enable the load last value when program start

true = load last stored value when start
false = do not load last value

- **Object Id**
  Bacnet Binary Value object ID
  1000 – 1199

- **In**
  Bacnet Binary Value object input value.

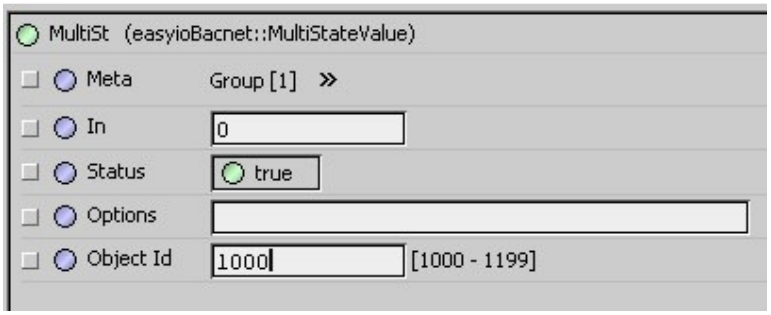


Example of using the BinaryValueRW , writing the PID loop enable. Noticed the bacnet object ID = 2000

4.5 *MultiStateValue*

MultiStateValue maximum 200 Multistate Value objects can be defined (ID: 1000 - 1199).

The property sheet of the object is shown below



- **In**
  Bacnet Multistate Value current input value.

- **Status**
  Bacnet Multistate Value object status. Readonly
  true = valid
  False = invalid

- **Options**
  Bacnet Multistate Value object option text, seperated by ;
  Maximum length: 50 characters
  Example:

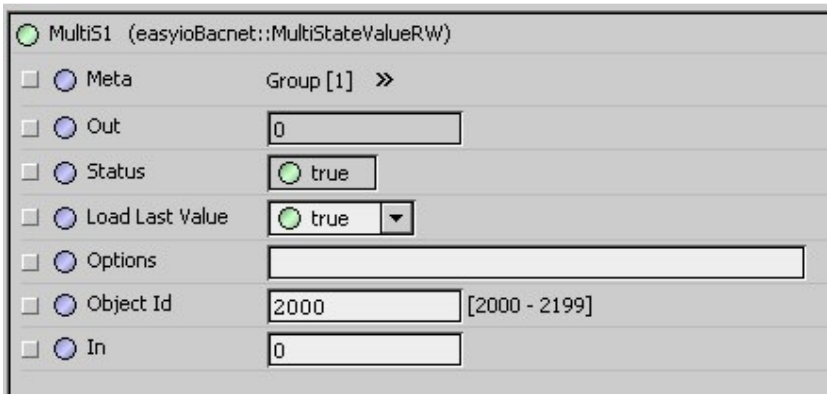
  ```
  voltage;Current;Resistance;Temperature;
  ```

- **Object ID**
  Bacnet Multistate Value object
  ID 000 - 1199

4.6 *MultiStateValueRW*

MultiStateValueRW maximum 200 Multistate ValueRW objects can be defined (ID: 2000 - 2199).

The property sheet of the object is shown below



- **Out**
  Bacnet Multistate Value current output value.

- **Status**
  Bacnet Multistate Value objects status. Readonly
  true = valid
  False = invalid

- **Load Last Value**
  Enable the load last value when program start
  true = load last stored value when start
  false = do not load last value

- **Options**
  Bacnet Multistate Value object option text, seperated by ;
  Maximum length: 50 characters
  Example:

  ```
  voltage;Current;Resistance;Temperature;
  ```

- **Object ID**
  Bacnet Multistate Value object
  ID 000 - 1199

5 EasyioComponent

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
5	EasyioComponent	1.0.43.10	Easyio 1.0.43.10 or higher	DayZone DEM5 DigitalStateTimer Drive FanControl Holiday MomentaryStartStop RTC SequenceLoop SingleLoop TimeZone

This kit contains 11 objects. All the objects are to be used for engineer the Sedona apps.

To use these objects just drag and drop into the wire sheet.

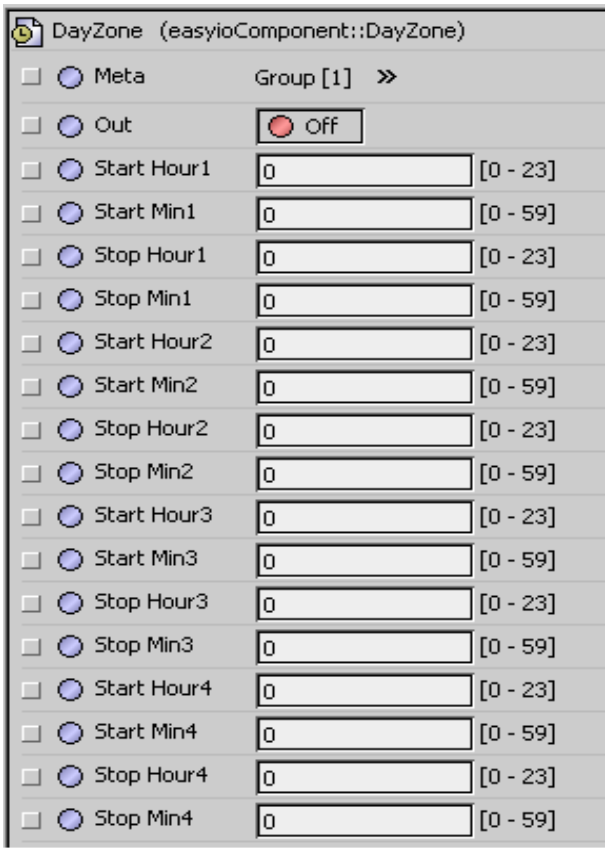
5.1 *DayZone*

DayZone component is used to specify the operation period of a day and each day zone consists of 4 periods specifying the start & stop time.

If the Start Time & Stop Time is set to 00:00, the period is considered all denied.

Usually implement together with **TimeZone** object, by linking its output to **TimeZone** object.

The property sheet of the object is shown below



- **Out**
  Boolean type of output (Readonly), indicate Current Day Zone state, set to ON when the controller current time is fall within one of the period of Day Zone.
  True = Day Zone Active
  False = Day Zone Inactive

- **Start Hour1**
  Period 1 start hour. Range from 0 – 23.

- **Start Min1**
  Period 1 start minute. Range from 0 – 59.

- **Stop Hour1**
  Period 1 stop hour. Range from 0 – 23.

- **Stop Min1**
  Period 1 stop minute. Range from 0 – 59.

- **Start Hour2**
  Period 2 start hour. Range from 0 – 23.

- **Start Min2**
  Period 2 start minute. Range from 0 – 59.

- **Stop Hour2**
  Period 2 stop hour. Range from 0 – 23.

- **Stop Min2**
  Period 2 stop minute. Range from 0 – 59.

- **Start Hour3**
  Period 3 start hour. Range from 0 – 23.

- **Start Min3**
  Period 3 start minute. Range from 0 – 59.

- **Stop Hour3**
  Period 3 stop hour. Range from 0 – 23.

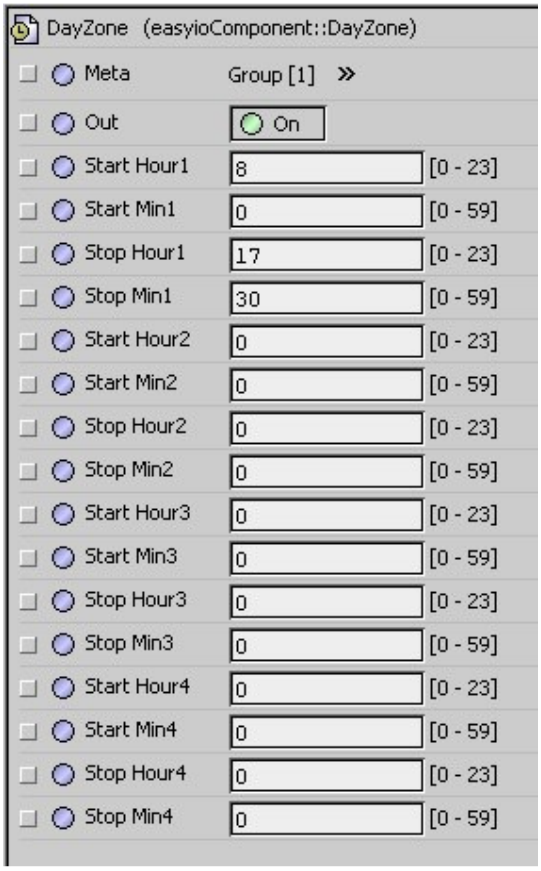
- **Stop Min3**
  Period 3 stop minute. Range from 0 – 59.

- **Start Hour4**
  Period 4 start hour. Range from 0 – 23.

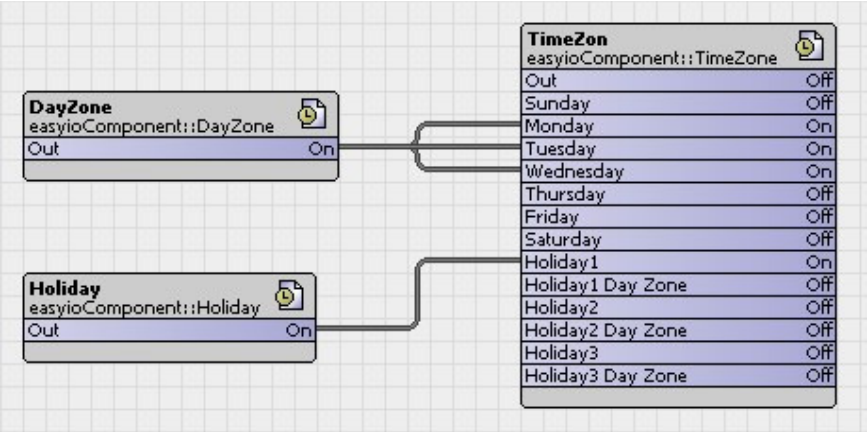
- **Start Min4**
  Period 4 start minute. Range from 0 – 59.

- **Stop Hour4**
  Period 4 stop hour. Range from 0 – 23.

- **Stop Min4**
  Period 4 stop minute. Range from 0 – 59.



Example of using the DayZone , Start time is set to 0800 hours and stop time is at 1730 hours .

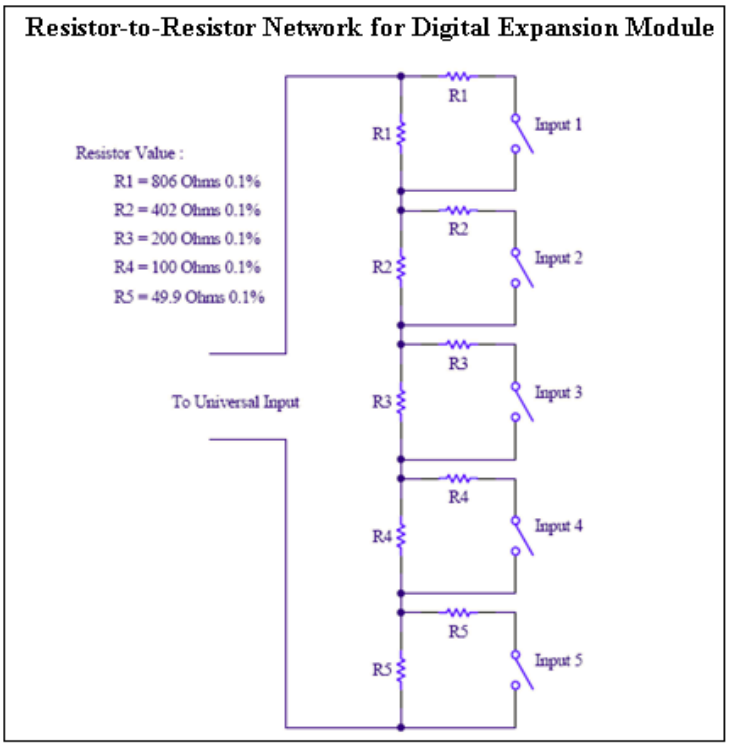


Example of using the DayZone with the TimeZone object .

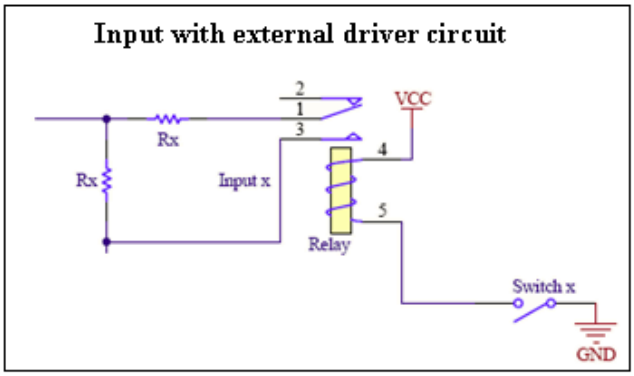
5.2 *DEM5*

DEM5 component provides up to 5 digital inputs through a single universal input (analog input) by using a specific resistor-to-resistor network connected to the universal input.

If EasyIO30P AI is used as input, the AI must be configured as Resistance 10K type (by setting the Analog Input Type to 5), refer to Analog Input component for setting.



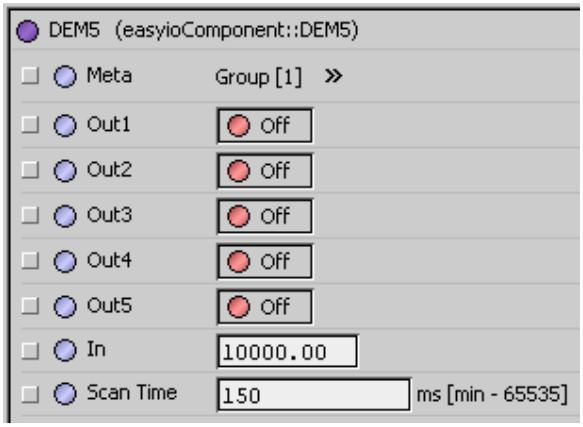
Only dry contracts (voltage free) with resistance less than 5 ohms can be used for the input. The resistance for the input open condition must be at least 1 mega ohms. For long wire connection or high resistance contact, external driver circuit is required as illustrated here.



The DEM5 digital inputs response time is about 2 seconds maximum. Due to the high resistor precision requirement, the EasyIO30P might not get the right resistance reading on the AI.

To compensate this, connect the DI Expander board with all input ON, and adjust the Analog Input value by alter the Analog Input Offset to about 778.

The property sheet of the object is shown below



- **Out1**
  Readonly. DEM5 digital output 1.

- **Out2**
  Readonly. DEM5 digital output 2.

- **Out3**
  Readonly. DEM5 digital output 3.

- **Out4**
  Readonly. DEM5 digital output 4.

- **Out5**
  Readonly. DEM5 digital output 5.

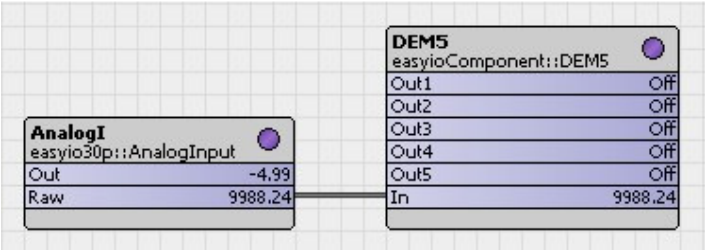
- **In**
  DEM5 analog input value.

- **Scan Time**
  Scan time of DEM5 processing, in milliseconds (ms). Maximum = 65535ms. Scan time of 500ms, means DEM5 will scan **In** every 500ms, and produce the 5 respective digital outputs.

Responses of 5 digital outputs with respect to **In** values:

Out1	Out2	Out3	Out4	Out5	In
Off	Off	Off	Off	Off	1545.44 - Max

Off	Off	Off	Off	On	1520.44 - 1545.43
Off	Off	Off	On	Off	1495.44 - 1520.43
Off	Off	Off	On	On	1470.44 - 1495.43
Off	Off	On	Off	Off	1445.44 - 1470.43
Off	Off	On	Off	On	1420.44 - 1445.43
Off	Off	On	On	Off	1395.44 - 1420.43
Off	Off	On	On	On	1369.94 - 1395.43
Off	On	Off	Off	Off	1343.44 - 1369.93
Off	On	Off	Off	On	1319.44 - 1344.43
Off	On	Off	On	Off	1294.44 - 1319.43
Off	On	Off	On	On	1269.44 - 1294.43
Off	On	On	Off	Off	1244.44 - 1269.43
Off	On	On	Off	On	1219.44 - 1244.43
Off	On	On	On	Off	1194.44 - 1219.43
Off	On	On	On	On	1168.44 - 1194.43
On	Off	Off	Off	Off	1142.44 - 1168.43
On	Off	Off	Off	On	1117.44 - 1142.43
On	Off	Off	On	Off	1092.44 - 1117.43
On	Off	Off	On	On	1067.44 - 1092.43
On	Off	On	Off	Off	1042.44 – 1067.43
On	Off	On	Off	On	1017.44 - 1042.43
On	Off	On	On	Off	992.44 - 1017.43
On	Off	On	On	On	966.94 - 992.43
On	On	Off	Off	Off	941.44 - 966.93
On	On	Off	Off	On	916.44 - 941.43
On	On	Off	On	Off	891.44 – 916.43
On	On	Off	On	On	866.44 – 891.43
On	On	On	Off	Off	841.44 - 866.43
On	On	On	Off	On	816.44 - 841.43
On	On	On	On	Off	791.44 - 816.43
On	On	On	On	On	Min - 791.43



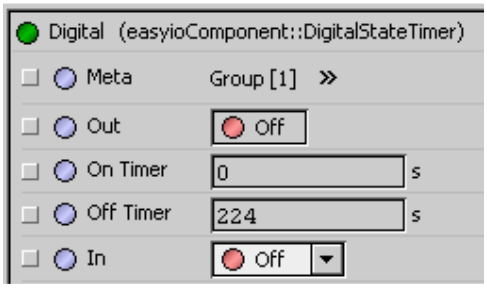
Example of using the DEM5 object. The DEM5 object must be linked from a analog input object

Image below show the wiring diagram for the DEM5

5.3 *DigitalStateTimer*

DigitalStateTimer is an object for On and Off period monitoring.

The property sheet of the object is shown below

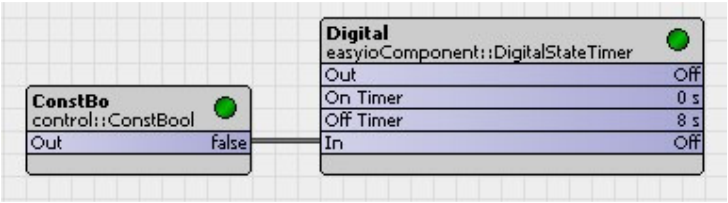


- **Out**
  Readonly output, which indicating Current digital state.

- **On Timer**
  Duration (in seconds) of the digital state remains in the ON state. The onTimer is reset to 0 automatically when an OFF to ON transition is occurred at the digital state and it will hold the value when the digital state is at OFF state. Use **resetOnTimer** to clear the timer manually.

- **Off Timer**
  Duration (in seconds) of the digital state remains in the OFF state. The offTimer is reset to 0 automatically when an ON to OFF transition is occurred at the digital state and it will hold the value when the digital state is at ON state. Use **resetOffTimer** to clear the timer manually.

- **In**
  Input digital state.

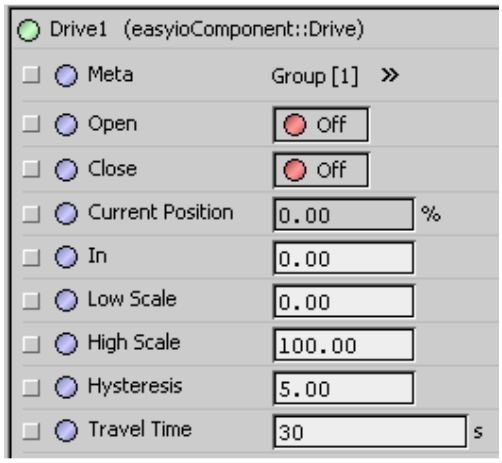


Example of using the Digital State Timer link from a Constant Boolean object.

5.4 *Drive*

Drive component provides the mechanism to drive a floating type actuator by using two outputs (**Open** and **Close** control). A single input with scale factor determines the desired position which controls the hardware output.
The open and close operation time is based on the full stroke travel time, **TravelTime**. When the calculated position hits the minimum (0%) or maximum (100%), the open or close output will continue run for the Drive Travel Time to make sure the actuator position is in place.

The property sheet of the object is shown below



- **Open**
  Readonly. Indicate Drive current open state.
  True = run
  False = stop

- **Close**
  Readonly. Indicate Drive current close state.
  True = run
  False = stop

- **Current Position**
  Current calculated position based on the Drive time.

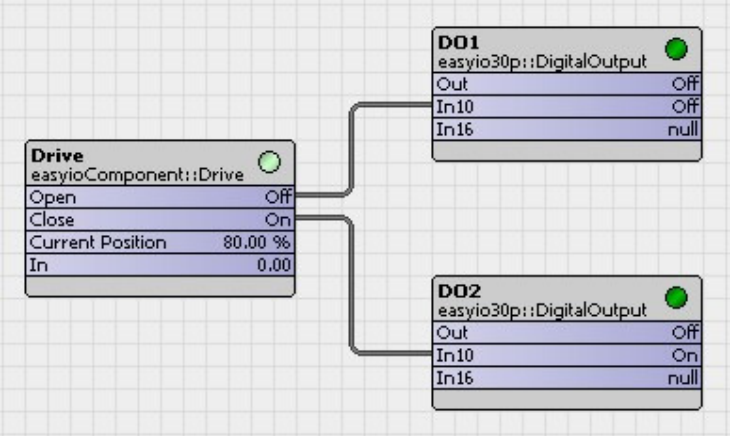
- **In**
  This parameter specifies the desired drive position. The input is scale to a range from 0% to 100% using the Drive **HighScale** and Drive **LowScale** parameters.

- **Low Scale**
  Defines the lowest value of Drive input value, which is equals to 0%. Default = 0.00.

- **High Scale**
  Defines the highest value of Drive input value, which is equals to 100%. Default = 100.00.

- **Hysteresis**
  Defines the minimum changes of the input value to activate the open and close operation. When the difference between input value and the calculated value is exceeding this limit, the Drive **Open** and Drive **Close** outputs will be activated to nullify the difference. Default = 5.00.

- **Travel Time**
  Drive full stroke travel time in seconds (s). Default value = 30s.

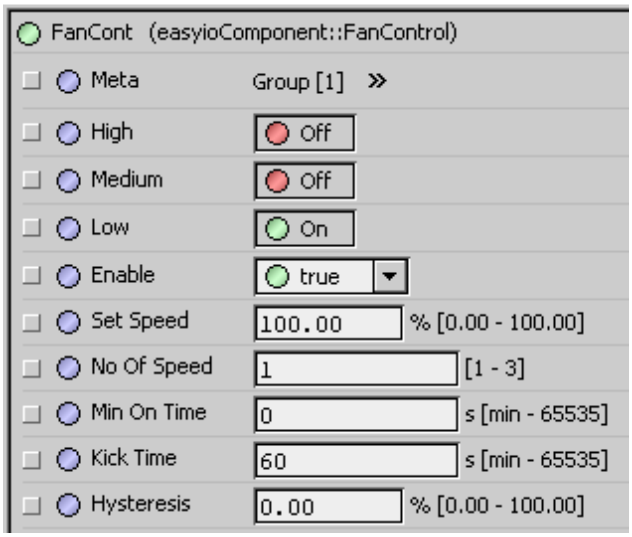


Example of using the Drive object controlling 2 Digital Output or a floating actuator

5.5 *FanControl*

FanControl component is able to drive a point-type output for sequenced control of up to three digital outputs to support one, two or three fan speed motor. The Fan Control also provides other control sequences such as minimum speed, minimum on time, kick time and hysteresis for a better fan operation.

The property sheet of the object is shown below



- **High**
  Readonly. Indicate the Fan Control digital high speed output state.
  True = ON, False = OFF

- **Medium**
  Readonly. Indicate the Fan Control digital medium speed output state.
  True = ON, False = OFF

- **Low**
  Readonly. Indicate the Fan Control digital low speed output state.
  True = ON, False = OFF

- **Enabled**
  Enable/disable the Fan Control function. If the Fan Control is disabled, all digital outputs will be set to off and all timers are reset.
  True = Enabled, False = Disabled

- **Set Speed**
  The desired speed in percentage (0 - 100%) used by the Fan Control to determine the output.

- **No Of Speed**
  To defines the Fan Control output type.
  1 = One Fan Speed (use Low Speed Output)
  2 = Two Fan Speed (use High and Low Speed Output)

3 = Three Fan Speed (use High, Medium and Low Speed Output)

The Fan Control output speed response to **NoOfSpeed** when **SetSpeed** changes as shown below:

Fan Control Output Speed

Type	Set Speed	Low Speed	Medium Speed	High Speed
One Fan Speed	0%	OFF	OFF	OFF
	>0%, <=100%	**ON**	OFF	OFF
Two Fan Speed	0%	OFF	OFF	OFF
	>0%, < 50%	**ON**	OFF	OFF
	>=50%, < 100%	OFF	OFF	**ON**
Three Fan Speed	0%	OFF	OFF	OFF
	>0%, < 33.33%	**ON**	OFF	OFF
	>=33.33%, < 66.66%	OFF	**ON**	OFF
	>=66.66%, <= 100%	OFF	OFF	**ON**

- **Min On Time**
  Defines the time period in seconds the fan control should run before it can be turned off. Default = 0s.

- **Kick Time**
  Defines the time period in seconds the fan control should run at highest speed when it starts to run from off state before it can be switched to the desired lower speed.

- **Hysteresis**
  The Hysteresis prevents the Fan Control point-type output to be changed too frequently, when the **SetSpeed** is swing at the margin value.

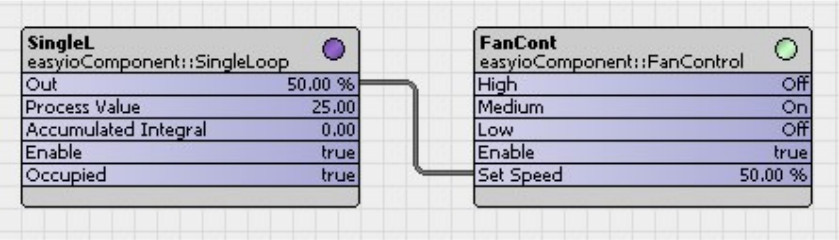
  Example: When the **SetSpeed** value at range 66.60% to 66.70%, the output will be switched in between **High** and **Medium**.

  Fan Control output speed response when **Hysteresis** is applied:

Fan Control Hysteresis

Type	Set Speed	Low Speed	Medium Speed	High Speed
One Fan Speed	<= (0% + H/2)	OFF	OFF	OFF
	> (0% + H), <=100%	**ON**	OFF	OFF
Two Fan Speed	<= (0% + H/2)	OFF	OFF	OFF
	>(0% + H), < (50% - H)	**ON**	OFF	OFF
	>(50% + H), <= 100%	OFF	OFF	**ON**
Three Fan Speed	<= (0% + H/2)	OFF	OFF	OFF
	> (0% + H), < (33.33% - H)	**ON**	OFF	OFF
	> (33.33% + H), < (66.66% - H)	OFF	**ON**	OFF
	> (66.66% + H), <= 100%	OFF	OFF	**ON**

where H = Hysteresis value

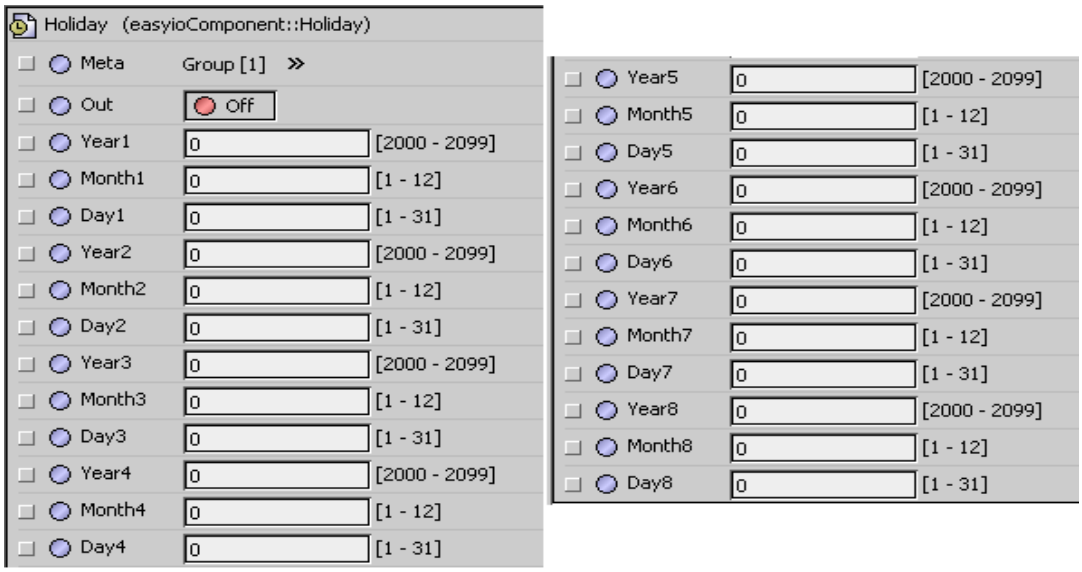


Example of using the FanControl object controlby a PID loop. The outputs can be link to Digital Output or virtual objects.

5.6 *Holiday*

Holiday object is used to specify up to 8 holiday dates. Usually implement together with **TimeZone**, by linking its output to **TimeZone's** holiday slots.

The property sheet of the object is shown below



- **Out**
  Readonly. The out state is set to ON when the controller current date equal to one of the date in the Holiday. On = Holiday, Off = Normal day

- **Year1**
  Holiday 1 Year's setting. Range from 2000 to 2099.

- **Month1**
  Holiday 1 Month's setting. Range from 1 to 12.

- **Day1**
  Holiday 1 Day's setting. Range from 1 to 31.

- **Year2**
  Holiday 2 Year's setting. Range from 2000 to 2099.

- **Month2**
  Holiday 2 Month's setting. Range from 1 to 12.

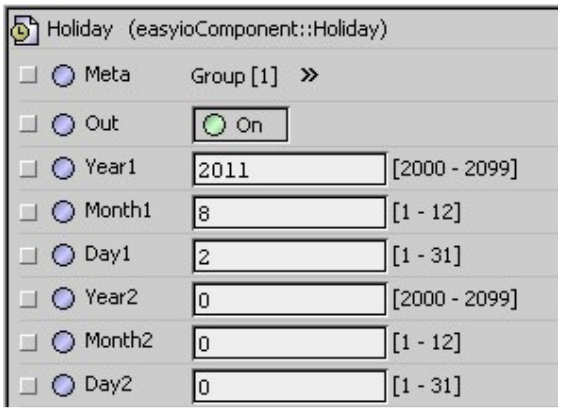
- **Day2**
  Holiday 2 Day's setting. Range from 1 to 31.

- **Year3**
  Holiday 3 Year's setting. Range from 2000 to 2099.

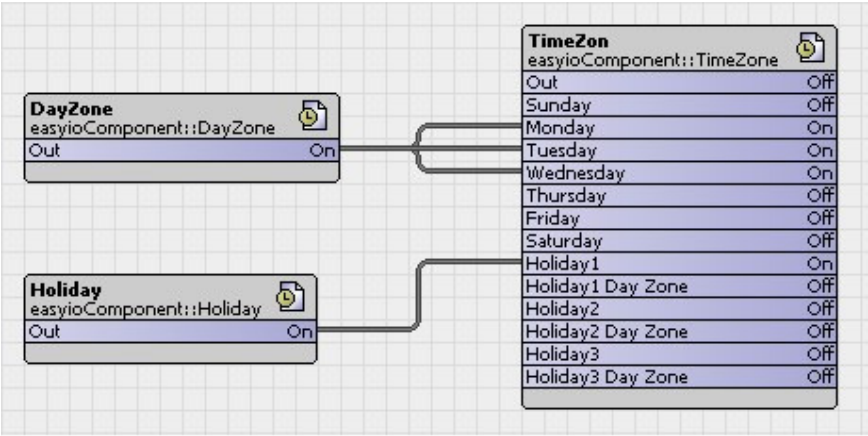
- **Month3**

Holiday 3 Month's setting. Range from 1 to 12.

- **Day3**
  Holiday 3 Day's setting. Range from 1 to 31
- **Year4**
  Holiday 4 Year's setting. Range from 2000 to 2099.
- **Month4**
  Holiday 4 Month's setting. Range from 1 to 12.
- **Day4**
  Holiday 4 Day's setting. Range from 1 to 31
- **Year5**
  Holiday 5 Year's setting. Range from 2000 to 2099.
- **Month5**
  Holiday 5 Month's setting. Range from 1 to 12.
- **Day5**
  Holiday 5 Day's setting. Range from 1 to 31
- **Year6**
  Holiday 6 Year's setting. Range from 2000 to 2099.
- **Month6**
  Holiday 6 Month's setting. Range from 1 to 12.
- **Day6**
  Holiday 6 Day's setting. Range from 1 to 31
- **Year7**
  Holiday 7 Year's setting. Range from 2000 to 2099.
- **Month7**
  Holiday 7 Month's setting. Range from 1 to 12.
- **Day7**
  Holiday 7 Day's setting. Range from 1 to 31
- **Year8**
  Holiday 8 Year's setting. Range from 2000 to 2099.
- **Month8**
  Holiday 8 Month's setting. Range from 1 to 12.
- **Day8**
  Holiday 8 Day's setting. Range from 1 to 31



Example of defining holiday date for the Holiday object.

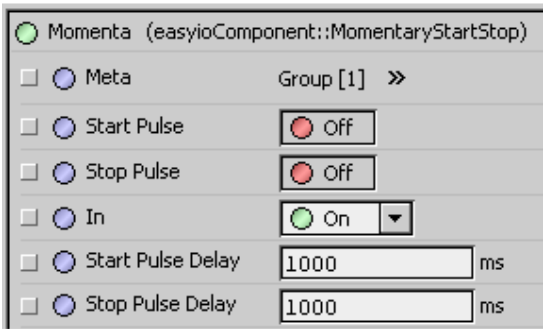


Example of using the Holiday object. It is linked to the TimeZone object. The Holiday object state is "ON" which overwrite the TimeZone Output for the day "Tuesday" which is "ON". The output is "OFF" when the holiday overwrite.

5.7 *MomentaryStartStop*

MomentaryStartStop provide StartPulse/StopPulse component provides the **StartPulse** and **StopPulse** pulse control to equipment that requires binary pulse on/off control rather than steady state signal.

The property sheet of the object is shown below



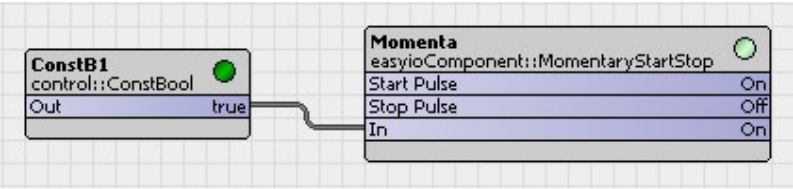
- **Start Pulse**
  Readonly start pulse output state. Turned On when **In** changed from Off to On; turn to Off after duration as configured in **Start Pulse Delay.**

- **Stop Pulse**
  Readonly stop pulse output state. Turned On when **In** changed from Off to On; turn to Off after duration as configured in **Stop Pulse Delay.**

- **In**
  On & Off digital input. On = Start Pulse, Off = Stop Pulse

- **Start Pulse Delay**
  Start pulse duration to hold on "On" state, in milliseconds. Default = 1000ms.

- **Stop Pulse Delay**
  Stop pulse duration to hold on "On" state, in milliseconds. Default = 1000ms.



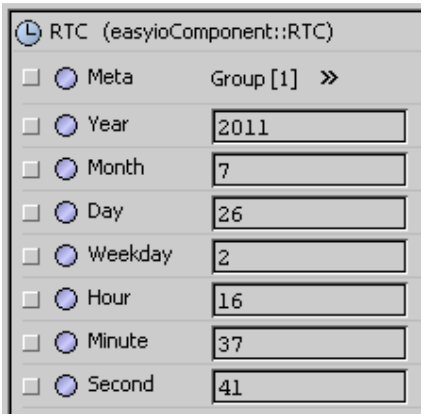
Example of using the Momentary Start Stop object. It is linked to a Boolean constant object.

5.8 *RTC (Real Time Clock)*

RTC is the Real Time Clock component showing the controller date & time information

The property sheet of the object is shown below

.



- **Year**
  Current year. Readonly

- **Month**
  Current month. Readonly

- **Day**
  Current day. Readonly

- **Weekday**
  Current day of week. Readonly

- **Hour**
  Current Hour. Readonly.

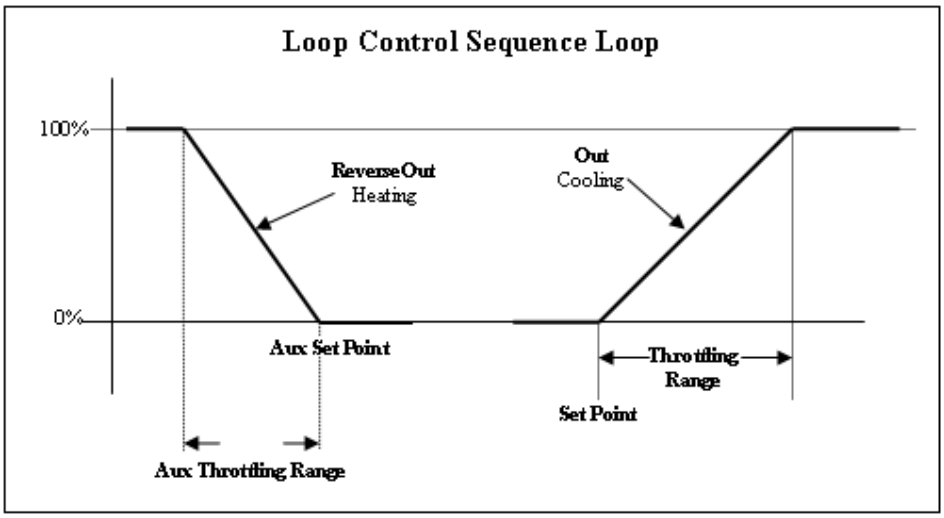
- **Minute**
  Current minute. Readonly.

- **Second**
  Current second. Readonly.

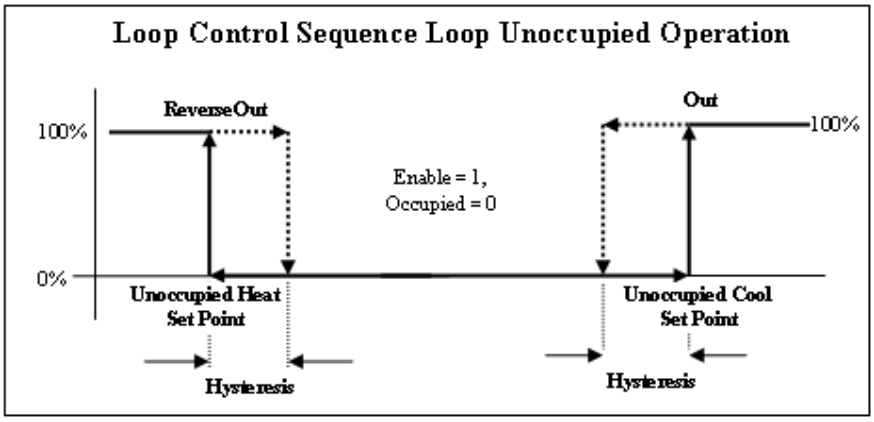
5.9 *SequenceLoop*

SequenceLoop is a component that provides proportional, integral and derivative (PID) control action of outputs based on the process value (input) and set point value. It monitors the process value, compare the process value to the set point, and calculate the output to reduce error (difference) between the set point and process value. The output is the result of proportional, integral and derivative calculation. This Sequence Loop consists of a single PID loop operation with two set points (Set Point and Aux Set Point) and two outputs (Output and Reverse/Aux Output).

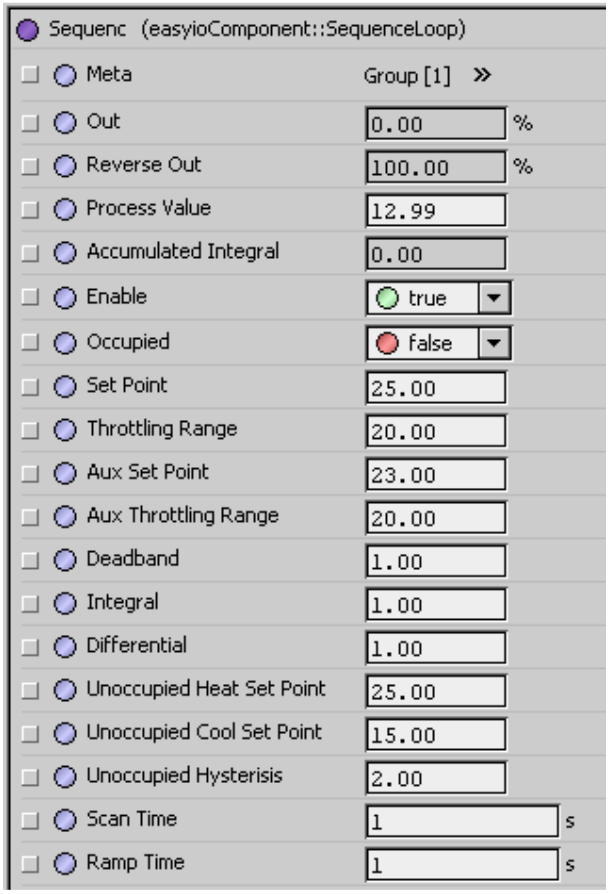
Typical sequence loop operation without the integral tuning and derivative tuning factor:



If loop control is disabled (Enable = 0), the loop control Out and ReverseOut will be set to 0. The Sequence Loop configuration operates the direct and reverse (cooling and heating) at the same time. They share the PID control algorithm and parameters except the set point and throttling range. This can be referred as auto mode since the loop control regulates the output by switching between the cooling and heating cycle. The SetPoint and ThrottlingRange are used for the cooling control and the Out as the output control. The AuxSetPoint and AuxThrottlingRange are used for the heating control and the ReverseOut as the output control. Under unoccupied setting (where Occupied is set to 0), cooling and heating operation are using the unoccupied set point and hysteresis setting.



The property sheet of the object is shown below



- **Out**

Readonly. Current calculated output value of the loop algorithm, in percentage.

- **Reverse Out**
  Readonly. Current calculated reverse output value of the loop algorithm in percentage.

- **Process Value**
  The loop control input value, normally derived from the analog input. This value will be used to compare with the **Set Point** value to determine **Output/Reverse Out** value.

- **Accumulated Integral**
  Readonly. This is the accumulated integral value over the loop process period if the Integral parameter is not zero.

- **Enabled**
  Enable the loop control operation.

- **Occupied**
  This parameter sets the occupied condition for loop control operation. Loop control uses different loop algorithm under occupied and unoccupied mode. False = Unoccupied, True = Occupied

- **Set Point**
  The control reference or the desired value used by loop algorithm.

- **Throttling Range**
  Defines the amount of input change required for loop control output to proportionally change from 0% to 100%.

- **Aux Set Point**
  The loop control secondary set point.

- **Aux Throttling Range**
  The loop control secondary throttling range.

- **Deadband**
  To Defines the minimum changes of **Process Value** for the loop control to take action.

- **Integral**
  Defines the loop integral gain parameter used in loop algorithm (Integral Tuning Parameter, I).

- **Differential**
  Defines the loop derivative gain parameter used in loop algorithm (Derivative Tuning Parameter, D).

- **Unoccupied Heat Set Point**
  The Heating operation set point for loop control during unoccupied mode.

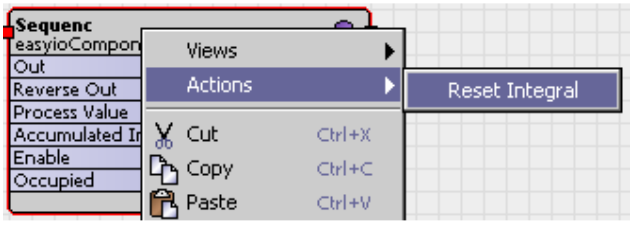
- **Unoccupied Cool Set Point**
  The Cooling operation set point for loop control during unoccupied mode.

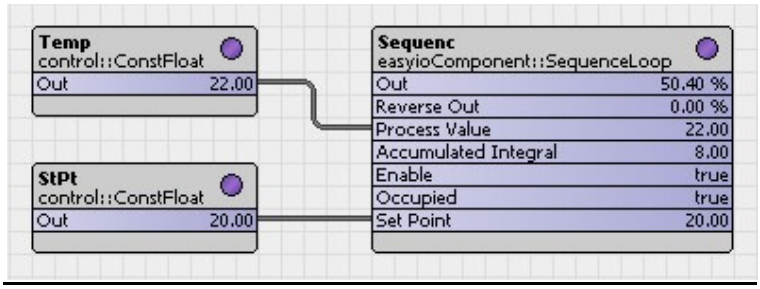
- **Unoccupied Hysterisis**
  The hysteresis control value of the Cooling and Heating operation during unoccupied mode.

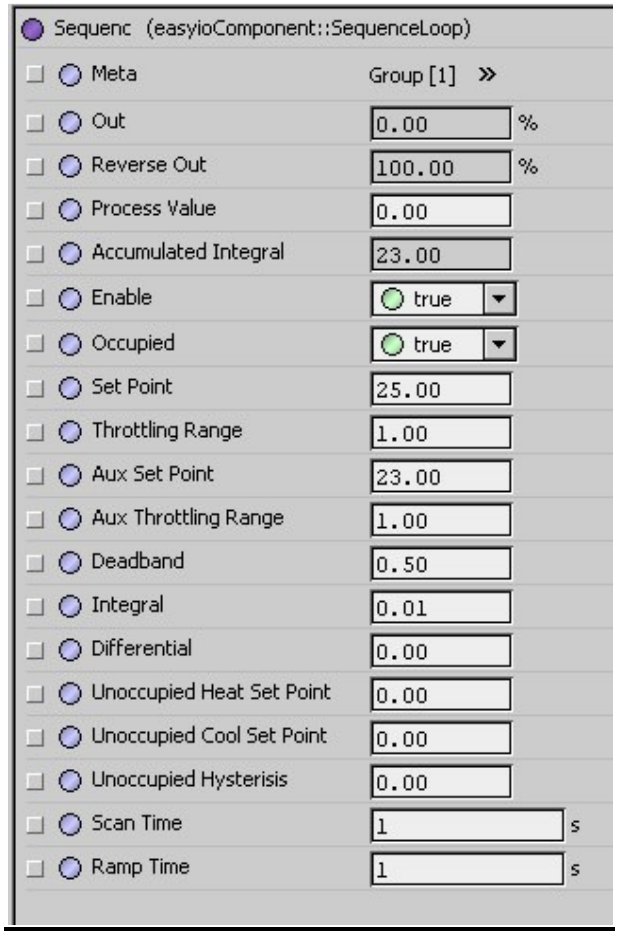
- **Scan Time**
  Defines the interval at which the process variable is sampled or the loop algorithm is executed in seconds (s).

- **Ramp Time**
  Defines the minimum interval at which the loop output may increase from 0% to 100% in seconds (s). This **Ramp Time** is only applied during the first start of loop operation and will be inhibiting when the output value reaching the desired output value.

- **Reset Integral**
  Manually reset the accumulated integral value (accumulated error value).





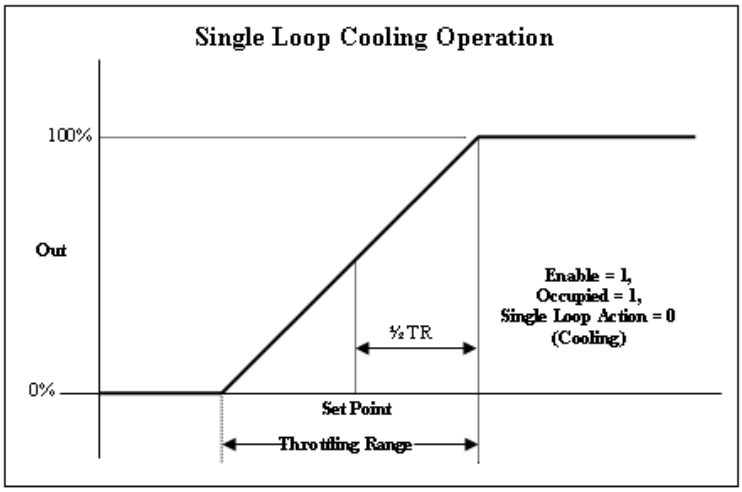


Example of using the Sequence Loop object.

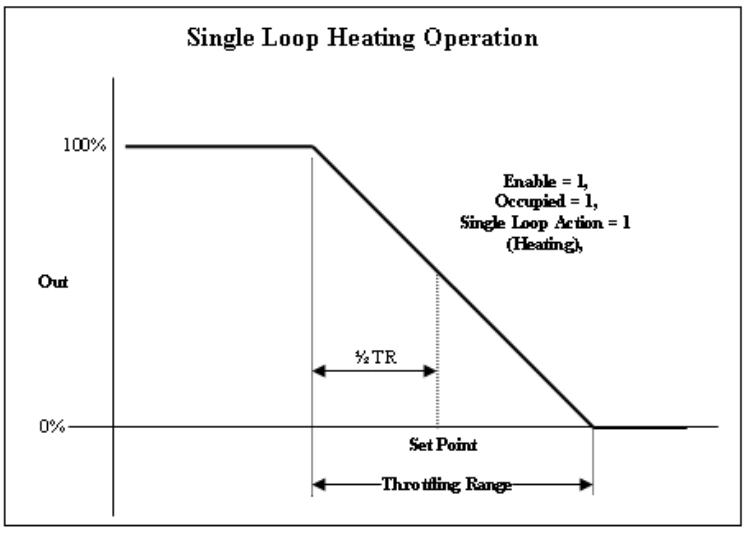
5.10 *SingleLoop*

SingleLoop is a component provides proportional, integral and derivative (PID) control action of outputs based on the process value (input) and set point value. It monitors the process value, compare the process value to the set point, and calculate the output to reduce error (difference) between the set point and process value. The output is the result of proportional, integral and derivative calculation. This single loop operation comes with single output and the output can be configured as either direct (cooling) or reverse acting. If only throttling range is applied to single loop operation (without integral and derivative tuning), the output is 50% when the input (Process Value) is equal to Set Point.

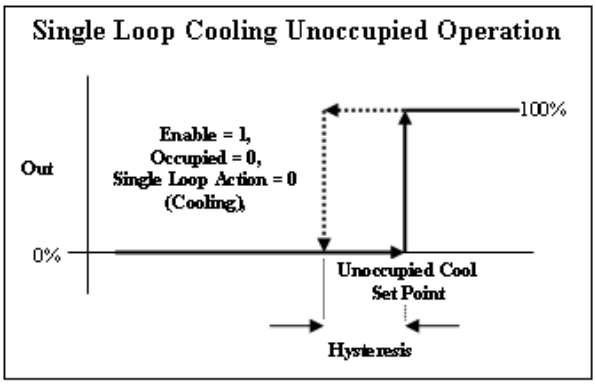
Typical cooling operation without the integral tuning and derivative tuning factor:

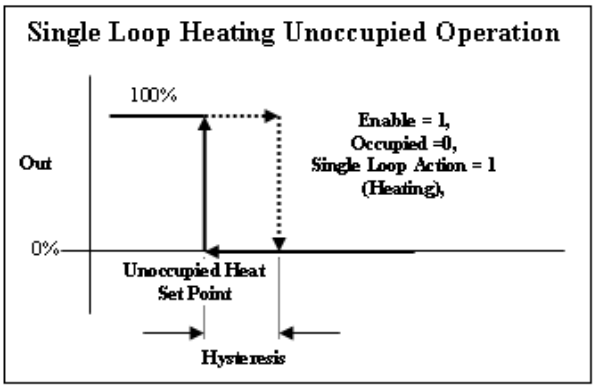


Typical heating operation without the integral tuning and derivative tuning factor:

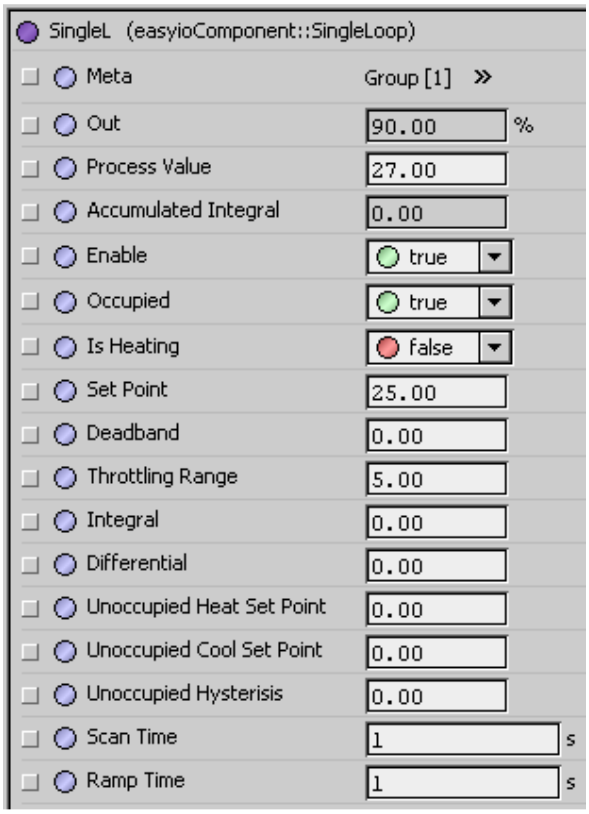


If loop control is disabled (Enable = 0), the Out will be set to 0. Under unoccupied setting (where Occupied is set to 0), cooling and heating operation are using the unoccupied set point and hysteresis setting. If the single loop is set for cooling under unoccupied mode, the output is set to 100% when the input is higher than the unoccupied cool set point. The output is set to 0% when the input is lower than the unoccupied cool set point minus hysteresis.





The property sheet of the object is shown below



- **Out**
  Readonly. Current calculated output value of the loop algorithm, in percentage.

- **Process Value**
  The loop control input value, normally derived from the analog input. This value will be used to compare with the SetPoint value to determine Output value.

- **Accumulated Integral**
  Readonly. This is the accumulated integral value over the loop process period if the Integral parameter is not zero.

- **Enable**
  Enable the loop control operation.

- **Occupied**
  This parameter sets the occupied condition for loop control operation. Loop control uses different loop algorithm under occupied and unoccupied mode. False = Unoccupied, True = Occupied

- **Is Heating**
  This parameter defines the output action of Single Loop configuration, either cooling or heating.
  False = Cooling, True = Heating

- **Set Point**
  The control reference or the desired value used by loop algorithm.

- **Deadband**
  To defines the minimum change of **Process Value** for the loop control to take action.

- **Throttling Range**
  Defines the amount of input change required for loop control output to proportionally change from 0% to 100%.

- **Integral**
  Defines the loop integral gain parameter used in loop algorithm (Integral Tuning Parameter, I).

- **Differential**
  Defines the loop derivative gain parameter used in loop algorithm (Derivative Tuning Parameter, D).

- **Unoccupied Heat Set Point**
  The Heating operation set point for loop control during unoccupied mode.

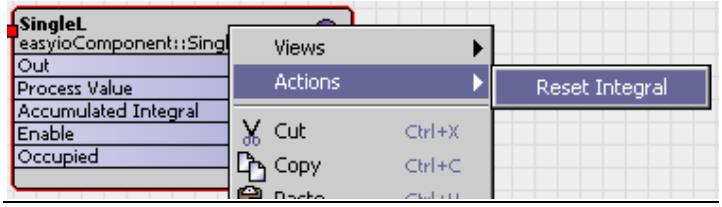
- **Unoccupied Cool Set Point**
  The Cooling operation set point for loop control during unoccupied mode.

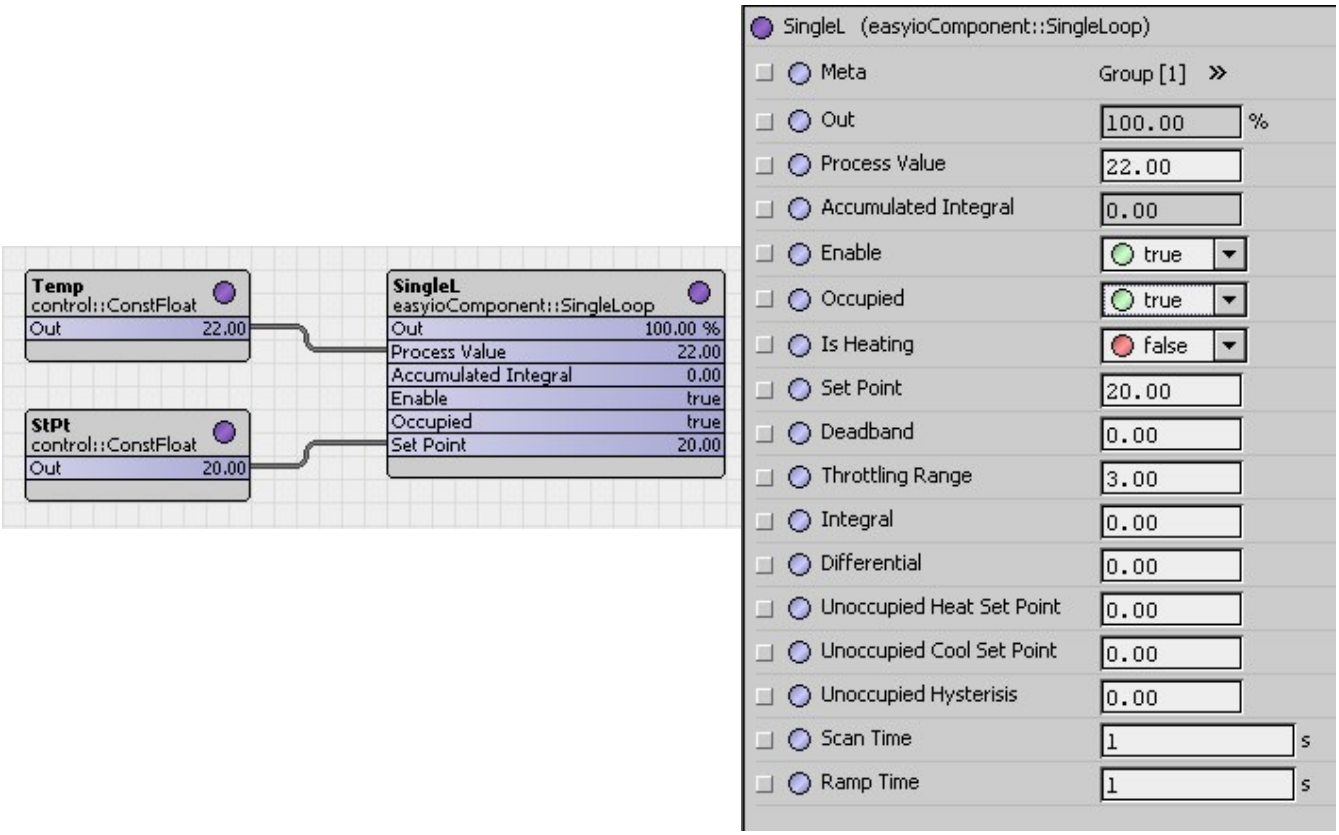
- **Unoccupied Hysterisis**
  The hysteresis control value of the Cooling and Heating operation during unoccupied mode.

- **Scan Time**
  Defines the interval at which the process variable is sampled or the loop algorithm is executed in seconds.

- **Ramp Time**
  Defines the minimum interval at which the loop output may increase from 0% to 100% in seconds. This RampTime is only applied during the first start of loop operation and will be inhibit when the output value reaching the desired output value.

- **Reset Integral**
  Manually reset the accumulated integral value (accumulated error value).



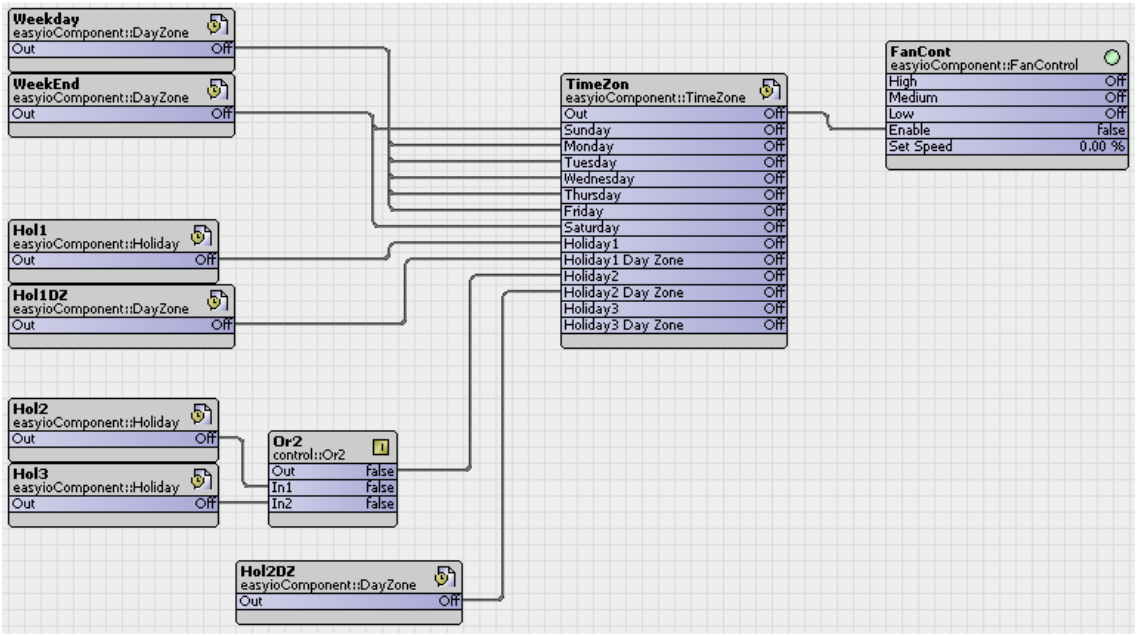


Example of using the Single Loop object.

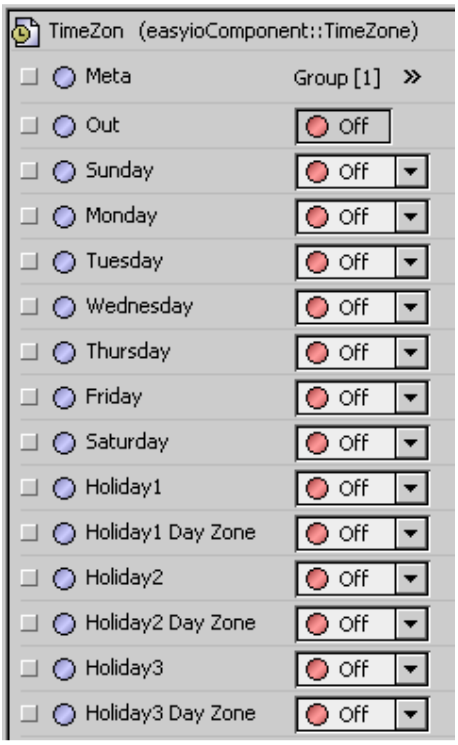
5.11 *TimeZone*

TimeZone object used to specify the weekly operation time including the holidays. Each time zone consists weekday (7 days, sunday to saturday) & 3 holidays day zone control. The Holiday List specifies the Holiday Schedule to be linked. The Holiday Lists have higher priority than the weekday setting.

Scheduler can be constructed using Day Zone, Time Zone & Holiday components.



The property sheet of the object is shown below



- **Out**
  Readonly. Time Zone current state. True = Active, False = Inactive
  State overridden priority:
  Holiday1 > Holiday2 > Holiday3 > Monday-Sunday (depends on controller weekday)

- **Sunday**
  Set to On (true) if the Sunday is true and controller weekday is Sunday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Monday**
  Set to On (true) if the Monday is true and controller weekday is Monday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Tuesday**
  Set to On (true) if the Tuesday is true and controller weekday is Tuesday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Wednesday**
  Set to On (true) if the Wednesday is true and controller weekday is Wednesday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Thursday**
  Set to On (true) if the Thursday is true and controller weekday is Thursday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Friday**
  Set to On (true) if the Friday is true and controller weekday is Friday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Saturday**
  Set to On (true) if the Saturday is true and controller weekday is Saturday and none of the holiday is active. This weekday setting is usually linked to Day Zone out property. On = Active, Off = Inactive

- **Holiday1**
  The Time Zone is in holiday state when this property set to true and the Time Zone out is determined by this holiday day zone. This holiday setting is usually linked to Holiday out property.
  True = Active, False = Inactive

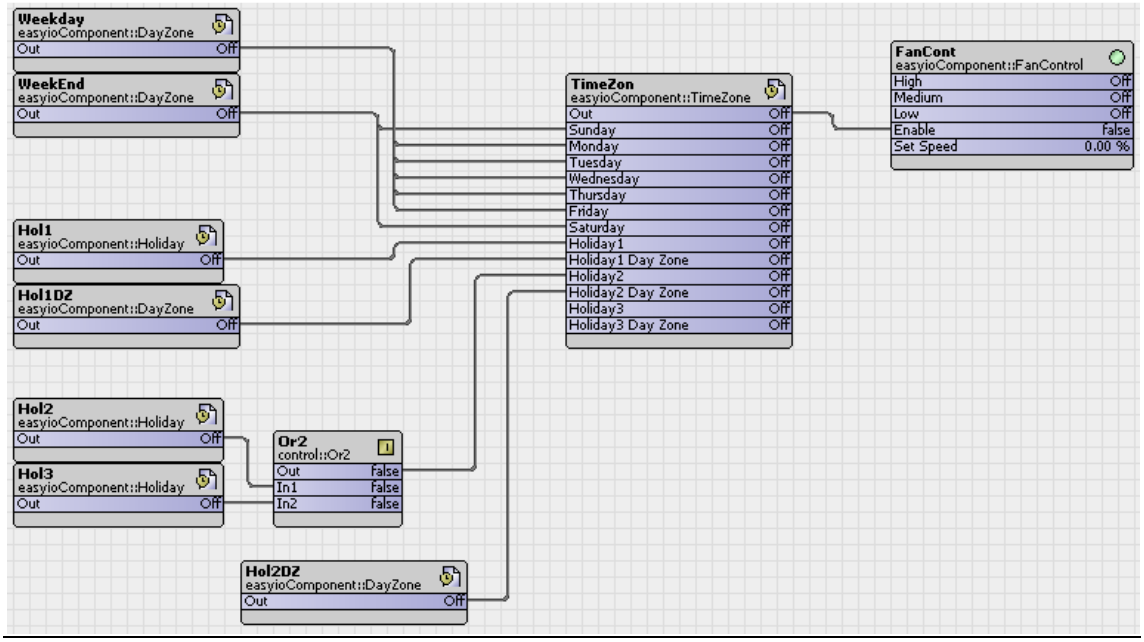
- **Holiday1 Day Zone**
  This property specifies the day zone control for holiday 1. This holiday day zone setting is usually linked to Day Zone out property.
  True = Active, False = Inactive

- **Holiday2**
  The Time Zone is in holiday state when this property set to true and the Time Zone out is determined by this holiday day zone. This holiday setting is usually linked to Holiday out property.
  True = Active, False = Inactive

- **Holiday2 Day Zone**
  This property specifies the day zone control for holiday 2. This holiday day zone setting is usually linked to Day Zone out property.
  True = Active, False = Inactive

- **Holiday3**
  The Time Zone is in holiday state when this property set to true and the Time Zone out is determined by this holiday day zone. This holiday setting is usually linked to Holiday out property.
  True = Active, False = Inactive

- **Holiday3 Day Zone**
  This property specifies the day zone control for holiday 3. This holiday day zone setting is usually linked to Day Zone out property.
  True = Active, False = Inactive



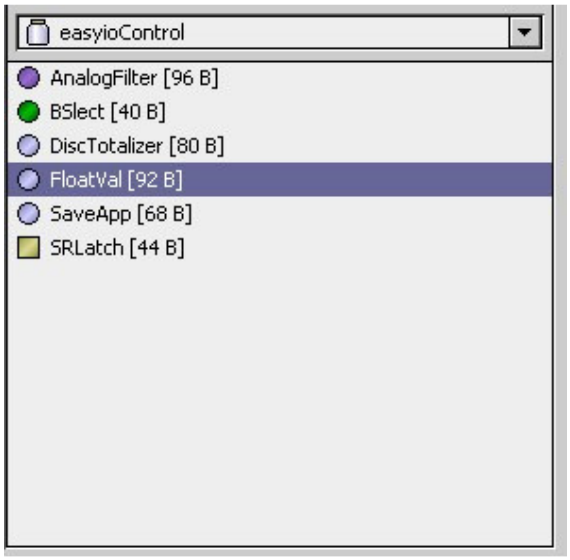
Example of using the Time Zone object. This object is combine with Day Zone and Holiday object

6 EasyioControl

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
6	EasyioControl	1.0.43.10	Easyio 1.0.43.00 or higher	AnalogFilter BSlect DiscTotalizer FloatVal Generic Table SaveApp

This kit contains 6 objects. All the objects are to be used for engineer the Sedona apps.

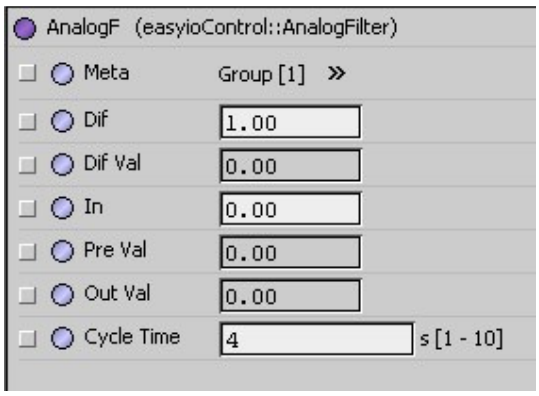
To use these objects just drag and drop into the wire sheet.



6.1 *AnalogFilter*

AnalogFilter object is use to limit a float value within a range. This is good for controlling a fluctuating float value where it always keeps hunting.

The property sheet of the object is shown below



- **Dif**
  User define offset value or cut off value.

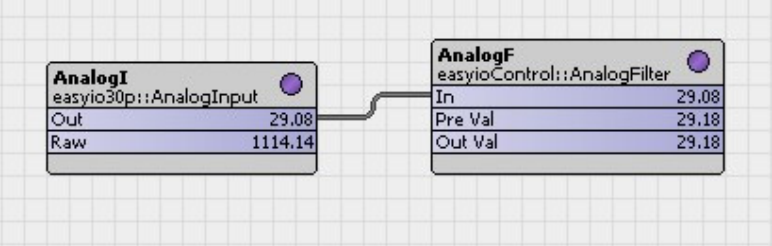
- **Dif Value**
  This is a read property where it represent a calculate different base on previous input.

- **In**
  Input float value to the object

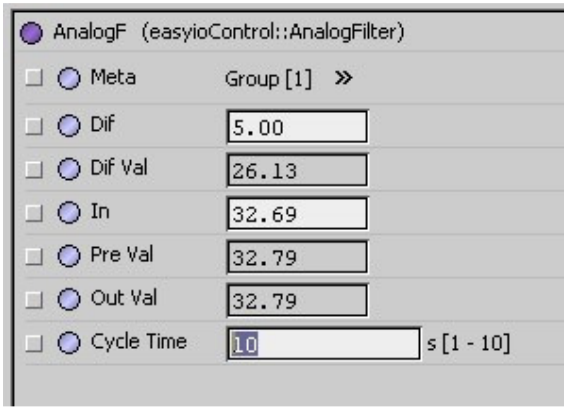
- **Pre Val**
  Last float value object received

- **Out Val**
  Output value after filtration

- **Cycle Time**
  Time period for the object to calculate and process the output.



Example of using the analog filter to slow down the poll rate of an 1.5K Platinum Temperature Sensor.

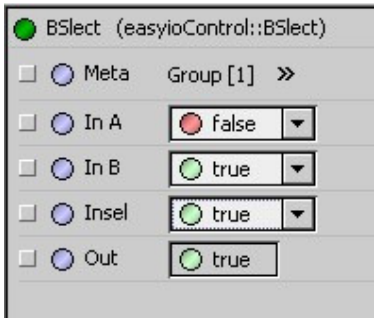


Example of property sheet for the Analog Filter object

6.2 *BooleanSelect*

BSlect is an object to select between 2 boolean inputs for the output value. The output value is also Boolean.

The property sheet of the object is shown below



- **In A**
  User define Input.

- **In B**
  User define Input

- **InSel**
  Selection of the Input.

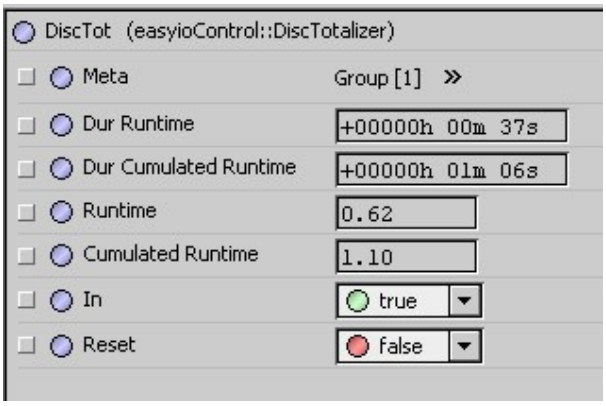
```
IF InSel = true
Out = In B

IF InSel = fasle
Out = In A
```

- **Out**
  Output of the selection depend on InSel

6.3 *DiscreteTotalizer*

DiscTotalizer is an object where you could totalize a Boolean value runtime.

The property sheet of the object is shown below



- **Dur Runtime**
  Duration runtime is the current runtime. The runtime will reset whenever there is a change in the input. Read Only

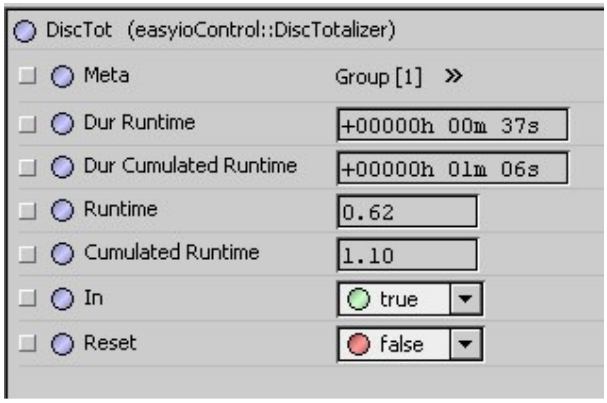
- **Dur Cumulated Runtime**
  Duration cumulated runtime is the total runtime. Read Only

- **Runtime**
  Current runtime in float value format in minute. Read Only

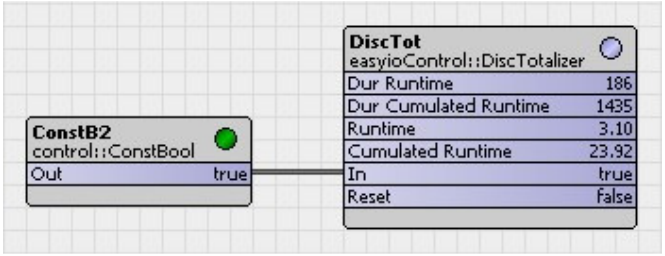
- **Cumulated Runtime**
  Duration cumulated runtime in float value format in minute. Read Only

- **In**
  Input for the Disctotalizer

- **Reset**
  To reset the counter



Example of the object while calculate the Runtime

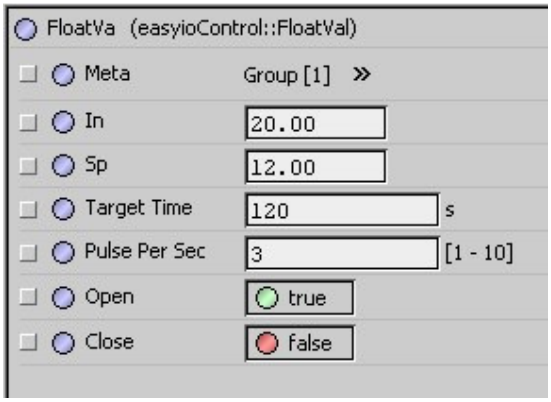


Example of the object in the wire sheet.

6.4 *FloatVal*

FloatVal is an object providing output as pulse depend on the input and setpoint.

The property sheet of the object is shown below



- **In**
  The input

- **Sp**
  Setpoint for the input

- **Target Time**
  Time period for the output “Open” or “Close” to be activated.

- **Pulse Per Sec**
  Pulse time period in seconds.

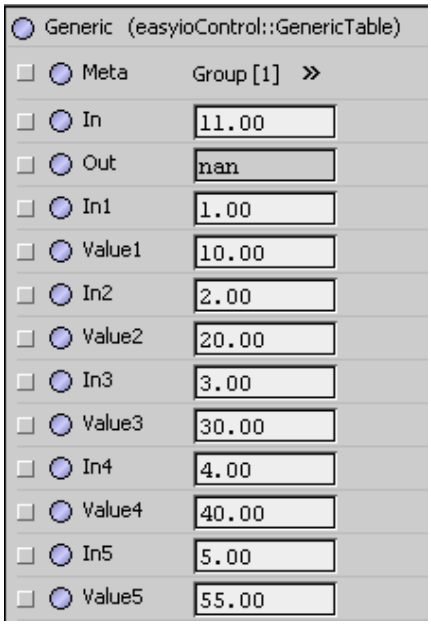
- **Open**
  Open Output

- **Close**
  Close output.

6.5 *Generic Table*

Generic Table is an object where it act like a curve fit. It can be used to tabulate a non-linear input.

The property sheet of the object is shown below

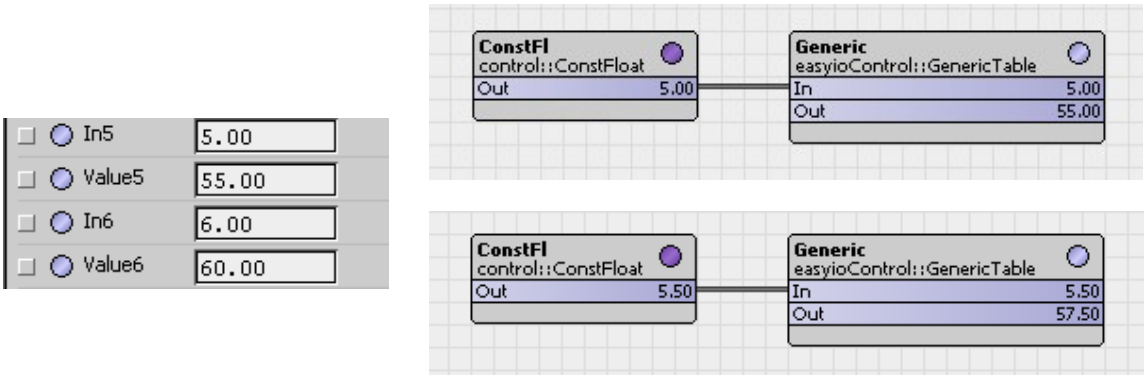


- **In**
  Input value for the generic table.

- **Out**
  Calculate value as per the table values key in.

- **In1 – In10**
  Input value 1 for the generic table.

- **Value1 – Value 10**
  Output Value base on the input 1 value.

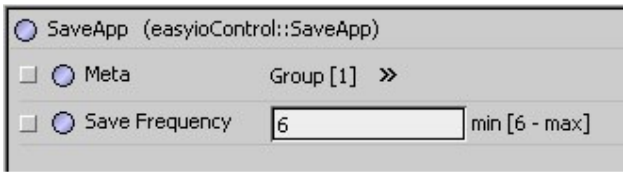


Example of a generic table object used

6.6 *SaveApp*

SaveApp is an object where it can used to automatically save the Sedona apps with a user pre-define period. Please do not set the frequency to be too low as there is a limited write cycle in the Flash Memory.

The property sheet of the object is shown below



- **Save Frequency**
  Period of time defines by user to automatically save the apps. Default is 6minutes.

7 EasyioDns

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
7	easyioDns	1.0.45.2	Easyio 1.0.43.00 or higher easyioLicense 1.0.45 or higher	DnsService

This kit contains 1 object. The object is to be used for retrieve the internet DNS host.

To use these objects just drag and drop into the wire sheet.

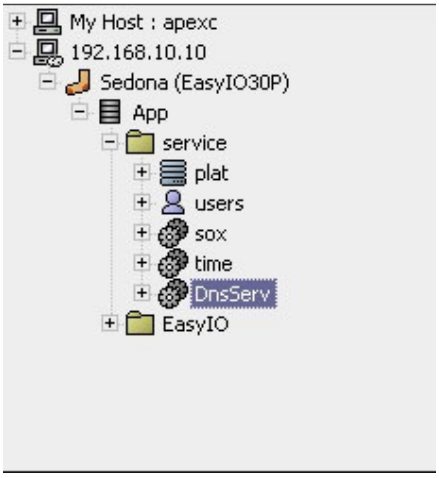
7.1 *DnsService*

DnsService is used to retrieve the IP address, corresponding to a given hostname.

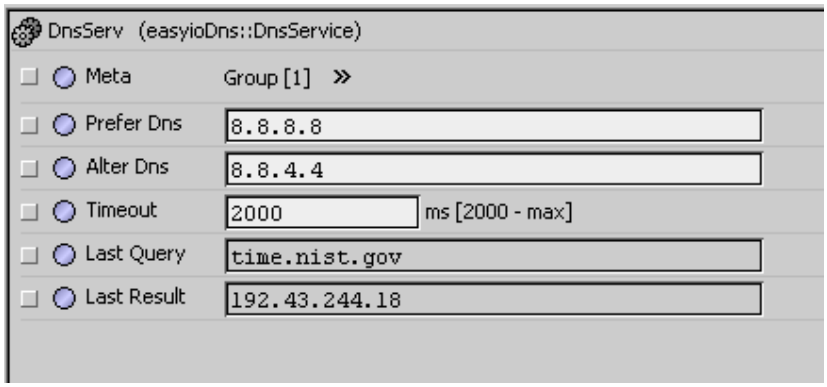
For example:
Hostname: www.google.com
IP address: 209.85.175.104

User will be needed to provide the hostname, and DnsService will return with the corresponding IP address.

****Note: DnsService must be drop inside Service folder (Sedona -> App -> Service).**



The property sheet of the object is shown below



- **Prefer Dns**
  IP address of Preferred DNS server used to lookup the IP address of the given hostname.

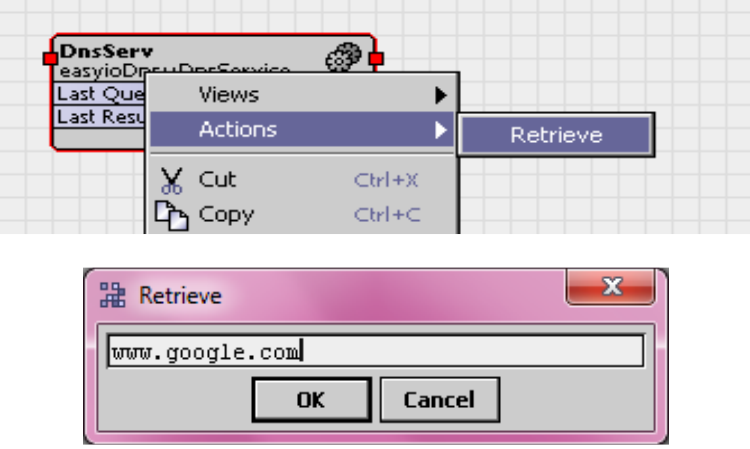
- **Alter Dns**
  IP address of Alternate DNS server, as a backup in the case that **Prefer Dns** not working.

- **Timeout**
  User defined time length to wait, before the DNS server giving a response, in milliseconds (ms). Default to its minimum value, 2000ms.

- **Last Query**
  The last query (hostname) sent to DNS server.

- **Last Result**
  The last response (corresponding to **Last Query**) receives from the DNS server.

- **Retrieve**
  Is use to test the internet connection. Any URL address will do.



Text input area allowed users to insert the hostname, which the IP address corresponding to this hostname will be retrieved through DNS server's response

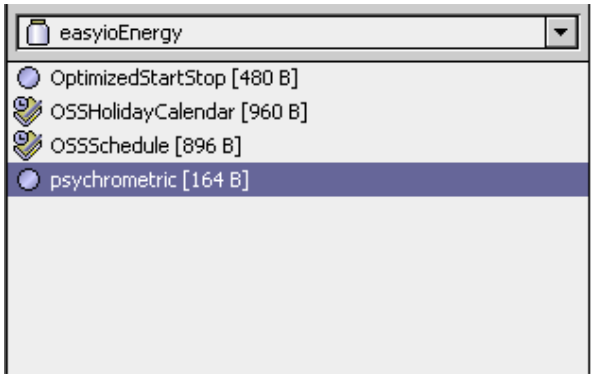
8 EasyioEmail

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
8	easyioEnergy	1.0.45.1	Easyio 1.0.43.10 or higher easyioFGLcd 1.0.45.4 or higher	OptimizeStartStop OSSHOlidayCalender OSSSchedule Psychometric

Easyioenergy kit is generally objects blocks for energy optimization. The working principle is very similar to Tridium Niagara Ax workbench optimize start stop object.

EasyioEnergy kit contains 3 objects :

To use these objects just drag and drop into the wire sheet.



9 EasyioEmail

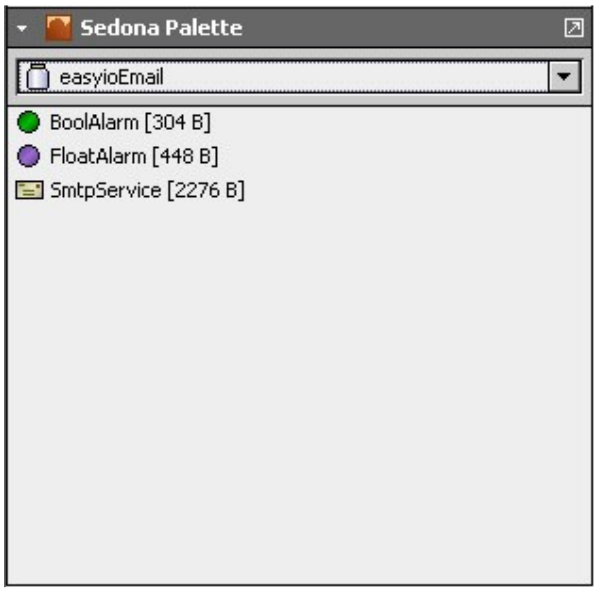
Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
9	easyioEmail	1.0.45.1	Easyio 1.0.43.10 or higher easyioDns 1.0.45 or higher easyioLicense 1.0.45 or higher	BoolAlarm FloatAlarm SmtpService

EasyioEmail kit is generally an alarm system, which embedded with SMTP(email) service. Once the alarm is triggered, an email will be sent to the recipients as configured.

At the moment only Boolean type and Float type is supported.

EasyioEmail kit contains 3 objects :

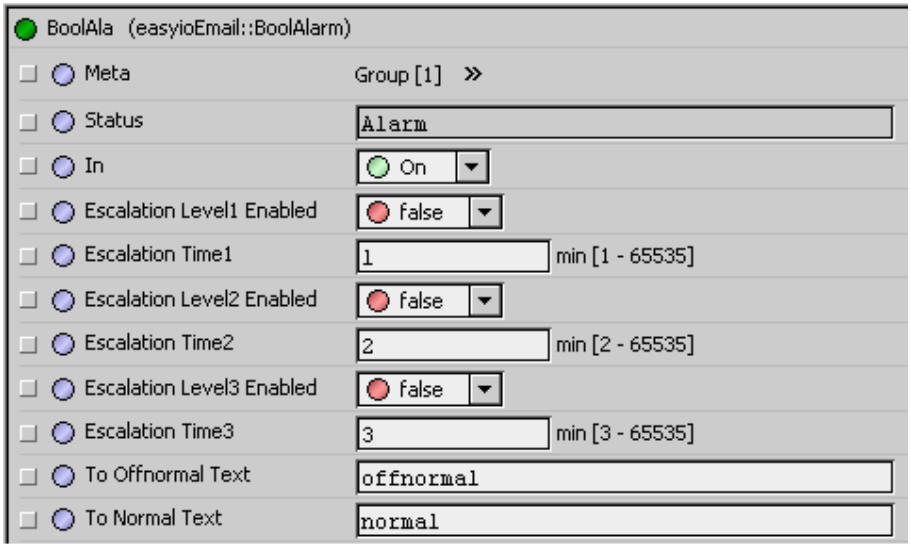
To use these objects just drag and drop into the wire sheet.



9.1 *BooleanAlarm*

BoolAlarm is an object to monitor Boolean type of data, which will trigger the alarm once the Boolean value changed from false to true.

The property sheet of the object is shown below



- **Status**
  To show current status of the **BoolAlarm**, indicated by either “Alarm” or “Normal”.

- **In**
  Input point with Boolean type. Alarm will be triggered once it changed from false to true (normal to offnormal).

- **Escalation Level1 Enabled**
  To enable or disable the level1 of alarm escalation, which will be set by user.

- **Escalation Time1**
  Delay time of the alarm escalation, in minute (min). 1 min means escalation alarm will occur 1 min after the first alarm. Default to its minimum value, 1 min.

- **Escalation Level2 Enabled**
  To enable or disable the level2 of alarm escalation, which will be set by user.

- **Escalation Time2**
  Delay time of the alarm escalation, in minute (min). 2 min means escalation alarm will occur 2 min after the first alarm. Default to its minimum value, 2 min.

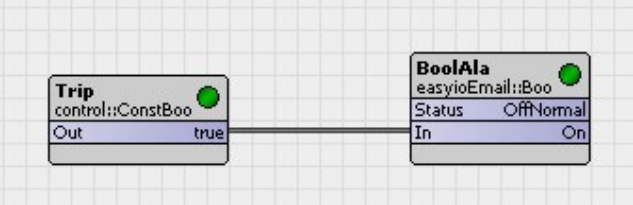
- **Escalation Level3 Enabled**
  To enable or disable the level3 of alarm escalation, which will be set by user.

- **Escalation Time3**

Delay time of the alarm escalation, in minute (min). 3 min means escalation alarm will occur 3 min after the first alarm. Default to its minimum value, 3 min.

- **To Offnormal Text**
  The text message to be sent as the information for recipients, as the Boolean input changed from normal to offnormal.

- **To Normal Text**
  The text message to be sent as the information for recipients, as the Boolean input changed from offnormal to normal.

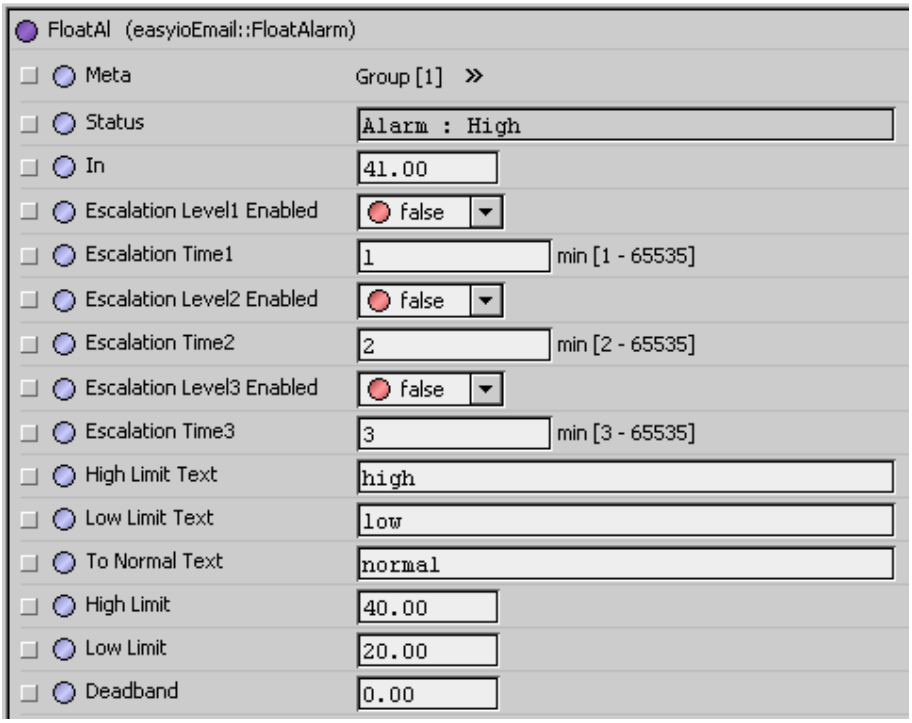


Example of configuring the Boolean point for Alarm Email

9.2 *FloatAlarm*

FloatAlarm is an object to monitor Float type of data, which will trigger the alarm once the Float value is out of its preset High or Low limit.

The property sheet of the object is shown below



- **Status**
  To show current status of the BoolAlarm, indicated by either “Alarm” or “Normal”.

- **In**
  Input point with Float type. Alarm will be triggered when its value out of either the preset High or Low limit.

- **Escalation Level1 Enabled**
  To enable or disable the level1 of alarm escalation, which will be set by user.

- **Escalation Time1**
  Delay time of the alarm escalation, in minute (min). 1 min means escalation alarm will occur 1 min after the first alarm. Default to its minimum value, 1 min.

- **Escalation Level2 Enabled**
  To enable or disable the level2 of alarm escalation, which will be set by user.

- **Escalation Time2**
  Delay time of the alarm escalation, in minute (min). 2 min means escalation alarm will occur 2 min after the first alarm. Default to its minimum value, 2 min.

- **Escalation Level3 Enabled**
  To enable or disable the level3 of alarm escalation, which will be set by user.

- **Escalation Time3**
  Delay time of the alarm escalation, in minute (min). 3 min means escalation alarm will occur 3 min after the first alarm. Default to its minimum value, 3 min.

- **High Limit Text**
  The text message to be sent as the information for recipients, as the Float input exceeds the High Limit.

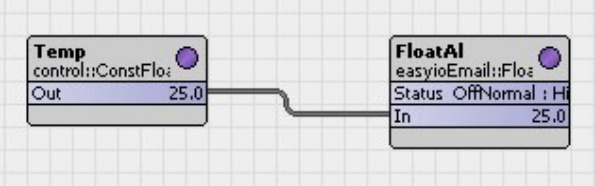
- **Low Limit Text**
  The text message to be sent as the information for recipients, as the Float input less than the Low Limit.

- **To Normal Text**
  The text message to be sent as the information for recipients, as the Float input fall between High and Low limit.

- **High Limit**
  Upper limit for the Float type input.

- **Low Limit**
  Low limit for the Float type input.

- **Deadband**
  High alarm = High Limit + Deadband
  Low alarm = Low Limit – Deadband

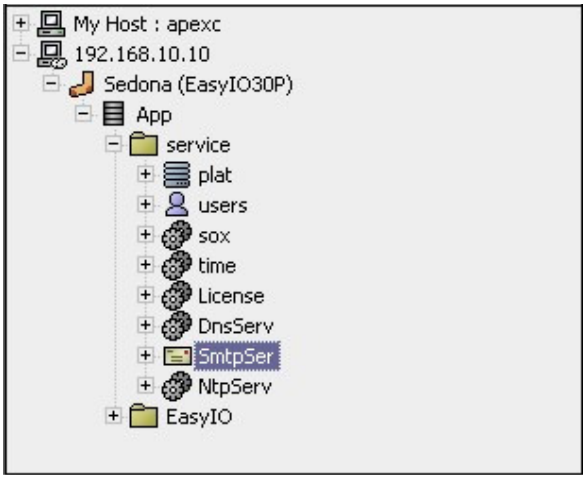


Example of configuring the Float point for Alarm Email

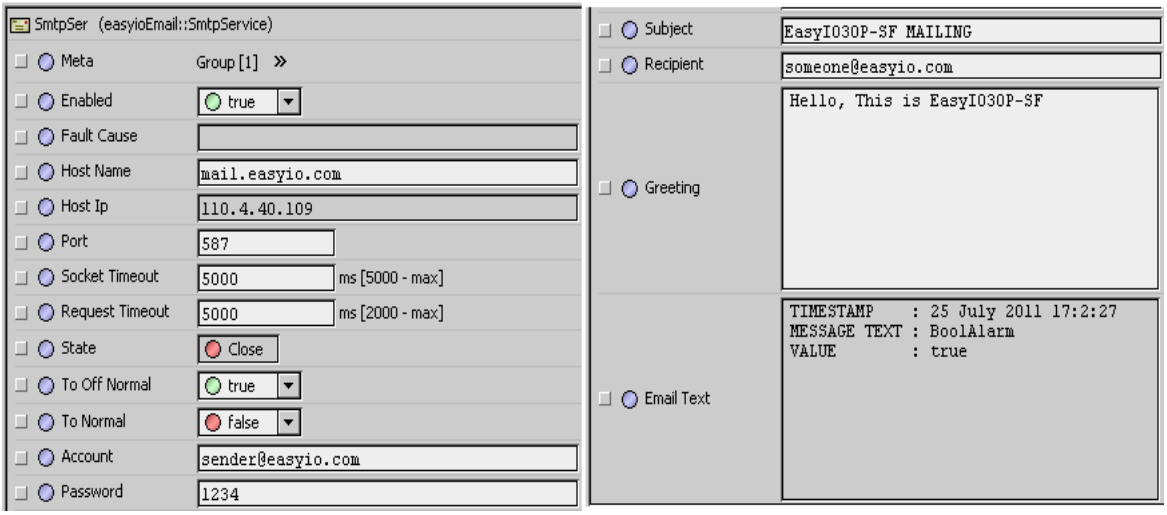
9.3 *SmtpService*

SmtpService is an object to provide the email service for the Alarm objects (BoolAlarm and FloatAlarm) mentioned above. When an alarm is triggered, **SmtpService** will send the message as configured, to the recipients.

****Note: SmtpService must be drop inside Service folder (Sedona -> App -> Service).**



The property sheet of the object is shown below



- **Enabled**
  To enable or disable the SmtpService, which will be set by user.

- **Fault Cause**
  To show cause of the error, when there was SmtpService failure.

- **Host Name**

The email server host name, for the SmtpService to send the email to recipients.

- **Host Ip**
  The IP address corresponding to the Host Name.

- **Port**
  The port number that used by SmtpService to send email.

- **Socket Timeout**
  User defined time length to wait, before the TCP Socket for communication is closed. Default to its minimum value, 5000ms.

- **Request Timeout**
  User defined time length to wait, before the SmtpService getting a response, in milliseconds (ms). Default to its minimum value, 5000ms.

- **State**
  State of the socket, whether Open or Close.

- **To Offnormal**
  To be set by user, whether true or false. If true, email will be sent when BoolAlarm/FloatAlarm changed from normal to offnormal, vice versa when set to false.

- **To Normal**
  To be set by user, whether true or false. If true, email will be sent when BoolAlarm/FloatAlarm changed from offnormal to normal, vice versa when set to false.

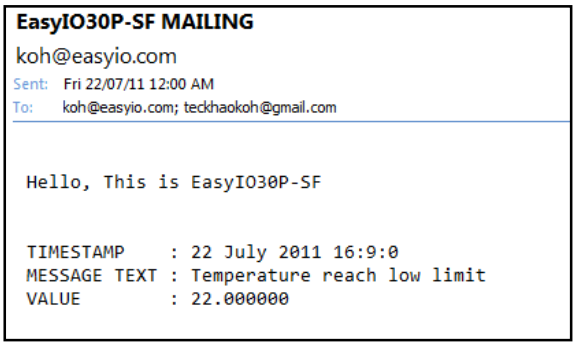
- **Account**
  Email account that will be used to send the email.

- **Password**
  Password of the email's **Account**.

- **Subject**
  Email's subject.

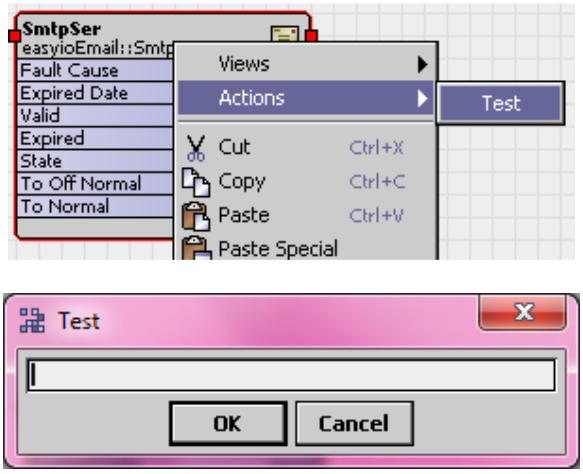
- **Recipient**
  Recipient(s) email address. Support multiple recipients by using semicolon (;) or comma (,).
  Example: abc@easyio.com;def@easyio.com
  abc@easyio.com,def@easyio.com

- **Greeting**
  The greeting words at the beginning of the email. User may customize the getting message to provide information such as device name and location.
  Example of email sent:



- **Email Text**
  The contents of last email sent, by SmtpService.

- **Test**



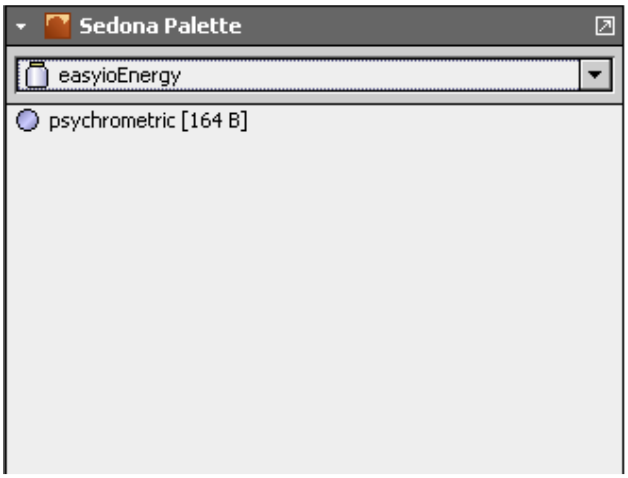
Text input area allowed users to enter email contents, to test the functionality of SmtpService. All properties of SmtpService have to be properly set before invoke this action.

10 EasyioEnergy

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
9	EasyioEnergy	1.0.45.1	Easyio 1.0.43.0	Psychometric

This kit contains 1 objects. These objects are used for energy control optimizations.

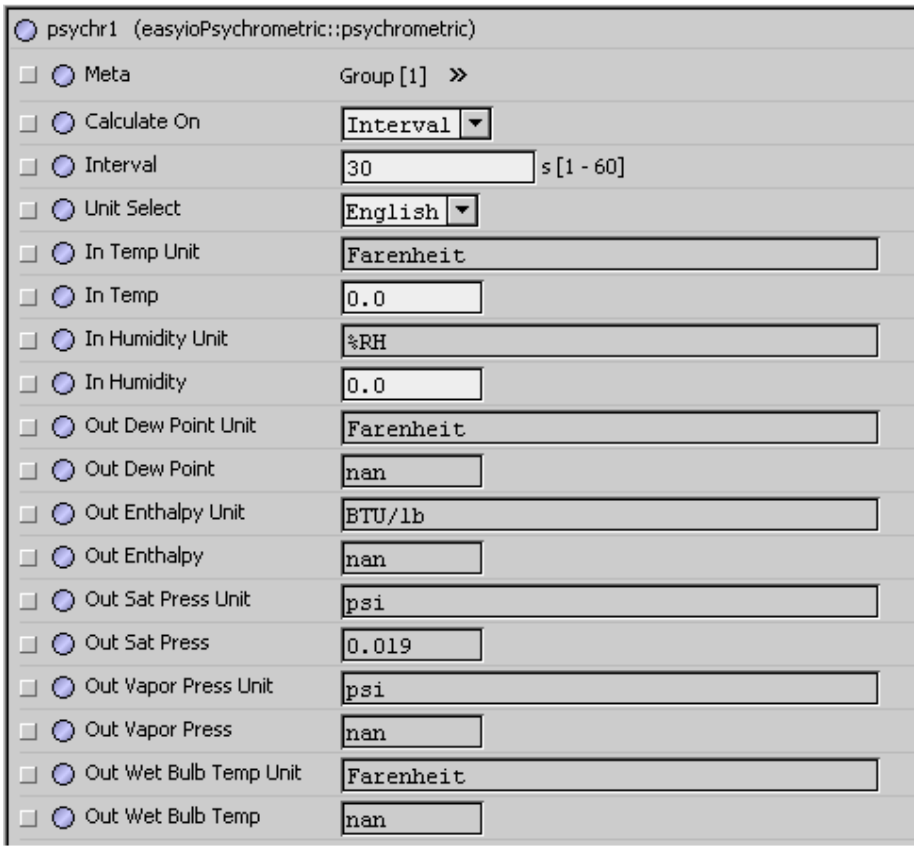
To use these objects, simply just drag and drop into the wire sheet.



10.1 *Psychometric*

Psychometric is a component calculating the psychometric values with the input of a Temperature and a Relative Humidity Value.

The property sheet of the object is shown below



- **Calculate On**
  Selection of calculation mode. If a COV is selected, Interval time period is no longer used.

- **Interval**
  Interval time period for the object to re-calculate. Only applicable if the Calculate On selection is selected to Interval.

- **Unit Select**
  Unit selection. Metric or English SI Units.

- **In Temp Unit**
  Input Temperature SI Unit

- **In Temp**
  Input Temperature Value

- **In Humidity Unit**
  Input Relative Humidity SI Unit

- **In Humidity**
  Input Relative Humidity Value

- **Out Dew Point Unit**
  Dew Point SI Unit. Read Only. Unit is reflection by the selection of the **Unit Select** Parameter

- **Out Dew Point**
  Calculated Dew Point Value.

- **Out Enthalpy Unit**
  Enthalpy SI Unit. Read Only. Unit is reflection by the selection of the Unit Select Parameter

- **Out Enthalpy**
  Calculated Enthalpy Value.

- **Out Saturation Pressure Unit**
  Saturation Pressure SI Unit. Read Only. Unit is reflection by the selection of the **Unit Select** Parameter

- **Out Saturation Pressure**
  Calculated Satturation Pressure Value.

- **Out Vaporize Pressure Unit**
  Vaporize Pressure SI Unit. Read Only. Unit is reflection by the selection of the **Unit Select** Parameter

- **Out Vaporize Pressure**
  Calculated Vaporize Pressure Value

- **Out Wet Bulb Unit**
  Wet Bulb SI Unit. Read Only. Unit is reflection by the selection of the **Unit Select** Parameter

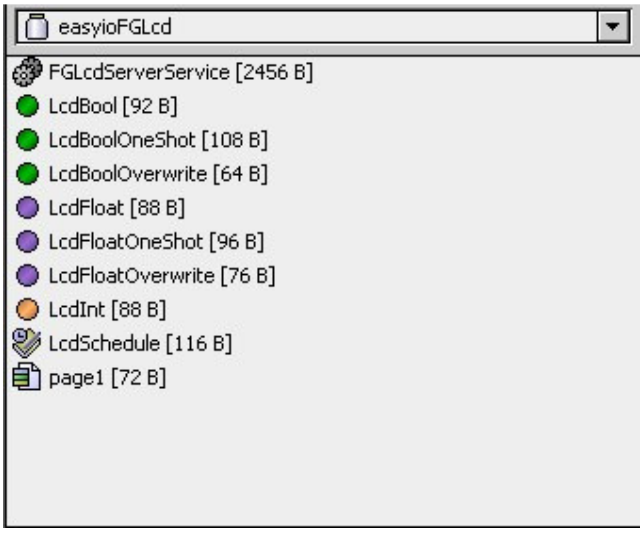
- **Out Wet Bulb**
  Calculated Wet Bulb Value.

11 easyioFGLcd kit

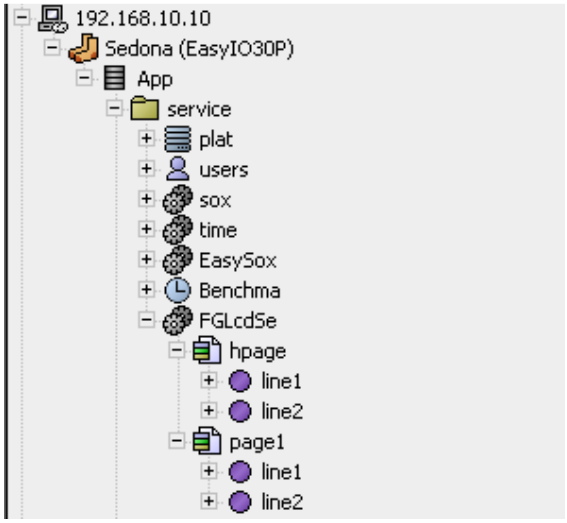
Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
9	EasyioFGLcd	1.0.45.1	Easyio 1.0.43.0	FGLcdServerService LcdBool LcdBoolOnShot LcdBoolOverwrite LcdFloat LcdFloatOnShot LcdFloatOverwrite LcdInt LcdSchedule Page1

This kit contains 10 object as show below.

To use this object just drag and drop into the wire sheet space.



The hierarchy of the LCD service is as below.

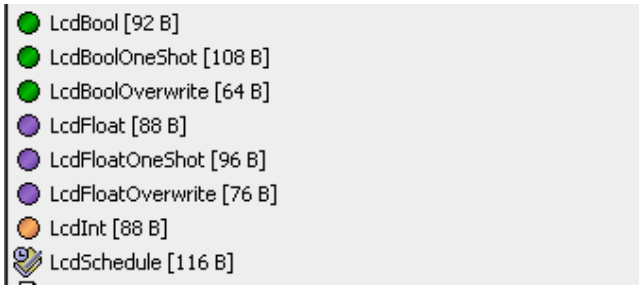


FGLcd Service object is a child of Service folder

Page object is a child of the FGLcd Service object

Line object is a child of the Pagex object.

Line objects are are objects as image below.

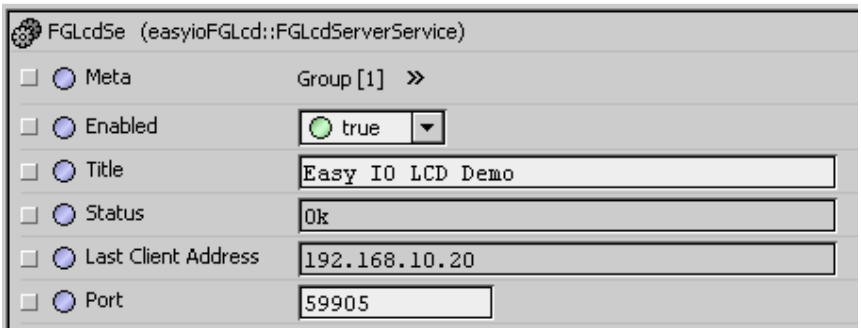


10.1 FGLcdServerService

FGLcdServerService is service object that drop in to the service folder wire sheet in order for the LCD to display values.

It is not necessary to drop the LCDService under the service folder. It can be anywhere in the apps.

The property sheet of the object is shown below



- **Enable**
  FGLcd service can be enable or disable with this parameter.

- **Title**
  This slot will be the display name of the controller when view at the LCD. It support max 13 characters.

- **Status**
  Status of the FGLcd Service object. It will show error when the LCD device might be having same port with other application. It will show "cannot bind to port."

- **Last Client Address**
  Shows the latest client connected to the server object. Client is the EasyIO LCD device.

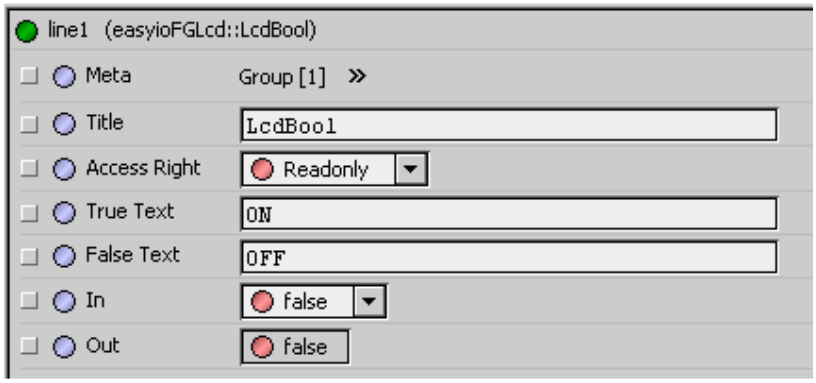
- **Port**
  By default the server port is 59905. However it is not supported with other port number at the moment.

10.2 LcdBool

LcdBool is a Boolean object and child object that sits under the ***"page"*** object. It is used to display Boolean point in the LCD display.

It can be a read-only object or writable object.

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

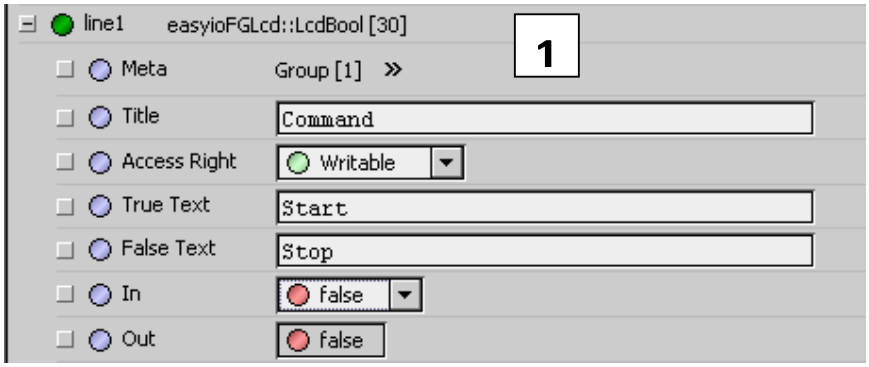
- **Access Right**
  Selection between Read-only or Writable.
  If writable is selected , title will be in **Bold** is view from the LCD

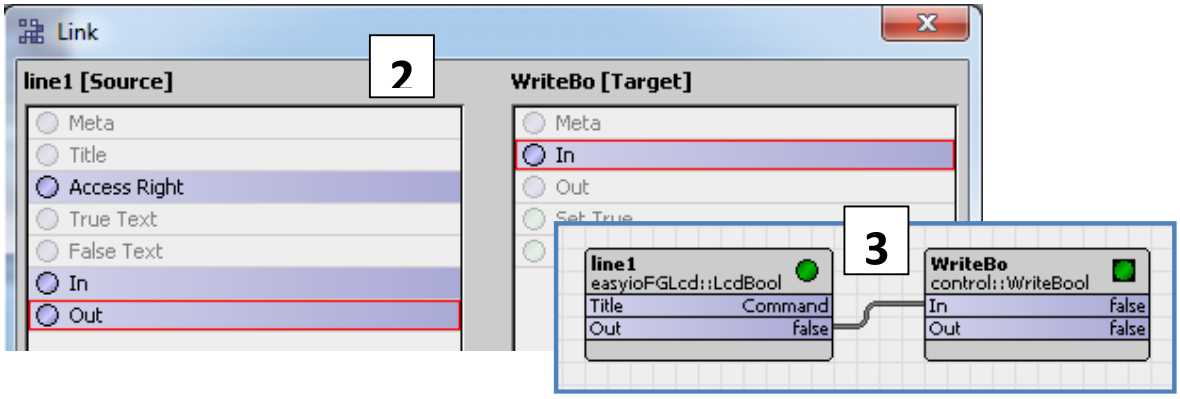
- **True Text**
  True text to be shown in the LCD display

- **False Text**
  True text to be shown in the LCD display

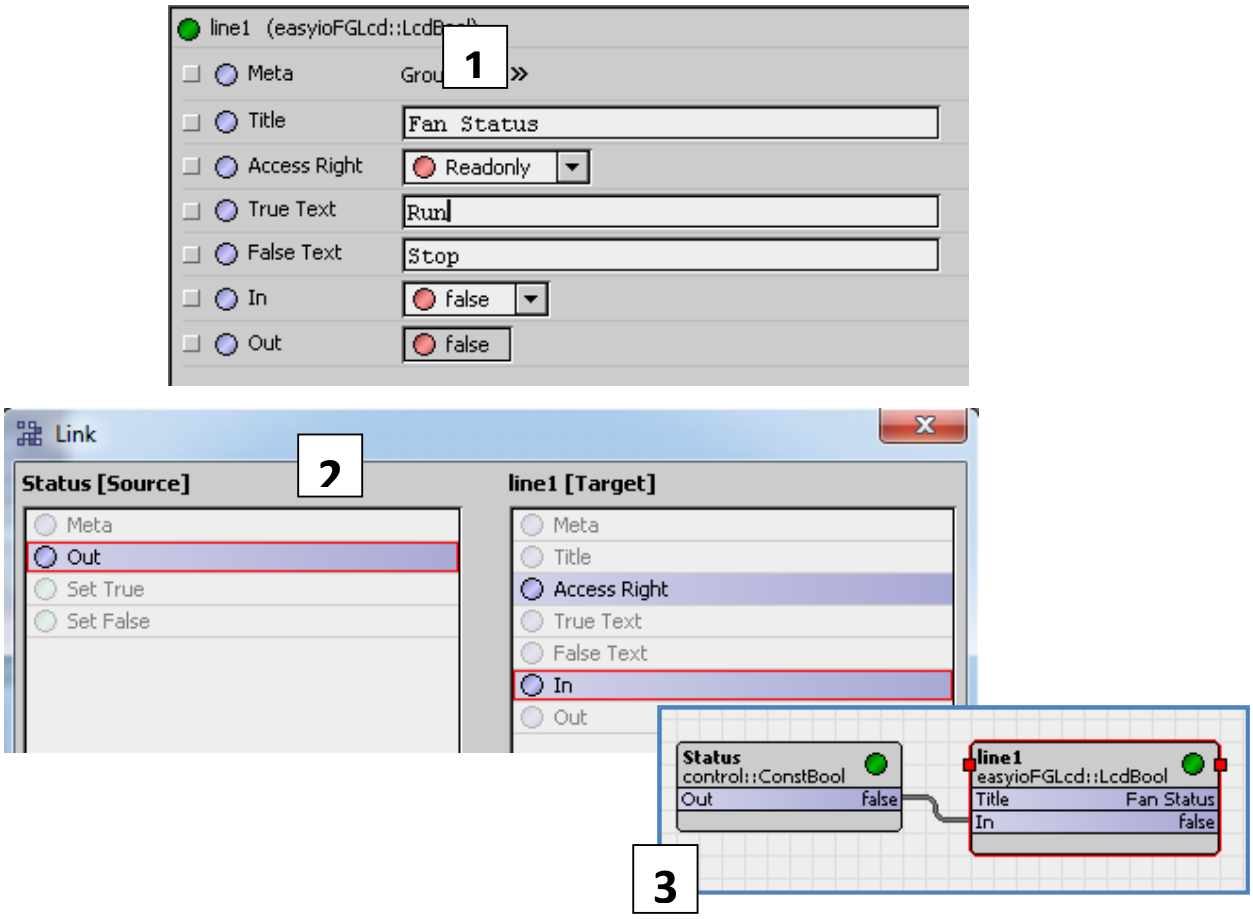
- **In**
  Input value of the object. When the object configured as read-only , input is link from another object within the Sedona apps.

- **Out**
  Output value of the object. When the object configured as writable, output is link to another object within the Sedona apps for control.





Example of configuring LcdBool as a writable point. Make sure you link to an object in the sedona apps that would be control by the LCD writable point.

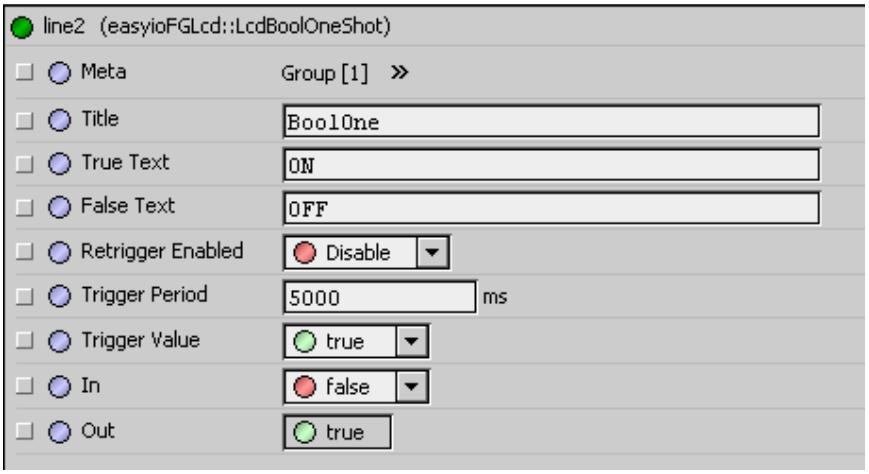


Example of configuring LcdBool as a read-only point. Make sure you link from an object in the sedona apps that would be monitor by the LCD read-only point.

10.3 LcdBoolOneShot

LcdBoolOneShot is a Boolean object and child object that sits under the ***"page"*** object. It is used to control Boolean point from the LCD display for a define time period.
It is a writable object.

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **True Text**
  True text to be shown in the LCD display

- **False Text**
  True text to be shown in the LCD display

- **Retrigger Enable**
  This property can be set to enable is re-trigger is required.

- **Trigger period**
  Period of the trigger period in seconds.

- **Trigger Value**
  Output Value when object is triggered.

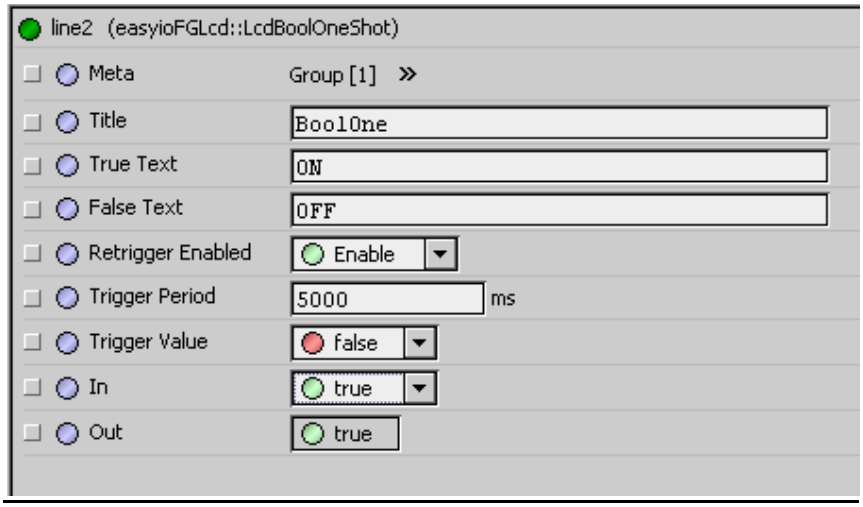
- **In**
  Input Constant value.

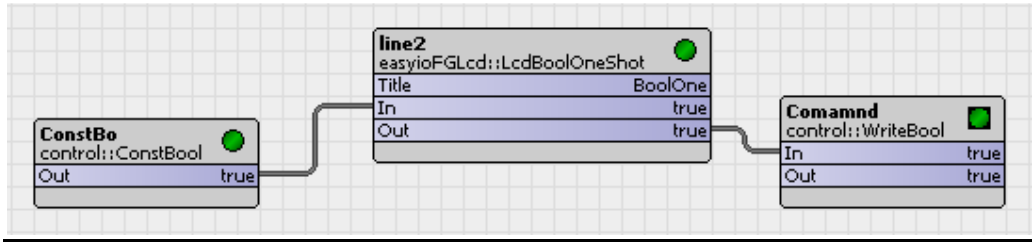
- **Out**
  Current Output Value. The control algorithm is a below.

  If the input is set to true , trigger value set to false , when a trigger fire

Output = false according to the period time.

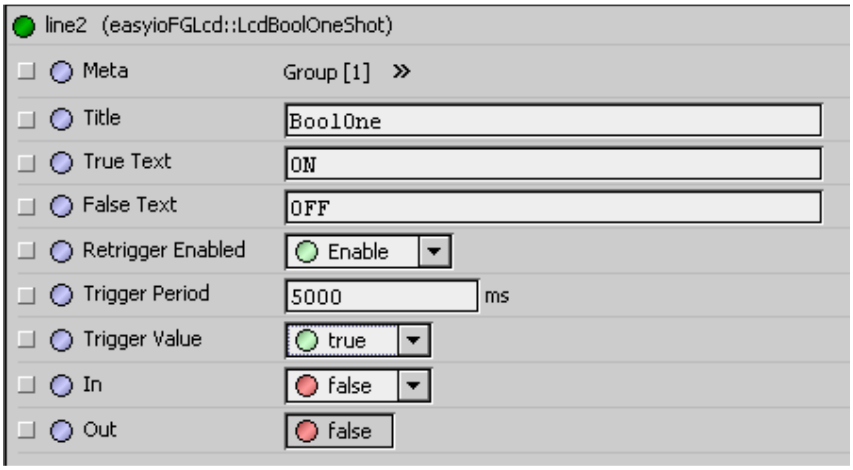
If the input is set to false , trigger value set to true , when trigger is fire
Output = true according to the period time.

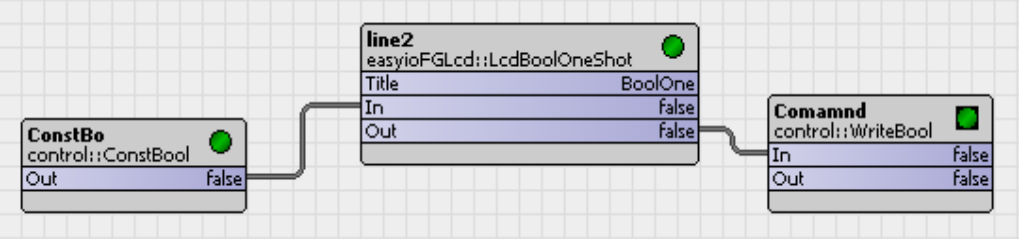




Example 01 of LcdBoolOneShoot,

In = true
Trigger value = false
Out = In ,
When object is trigger output = false for 5secs.





Example 02 of LcdBoolOneShoot,

In = false
Trigger value = true
Out = In ,
When object is trigger output =true for 5secs.

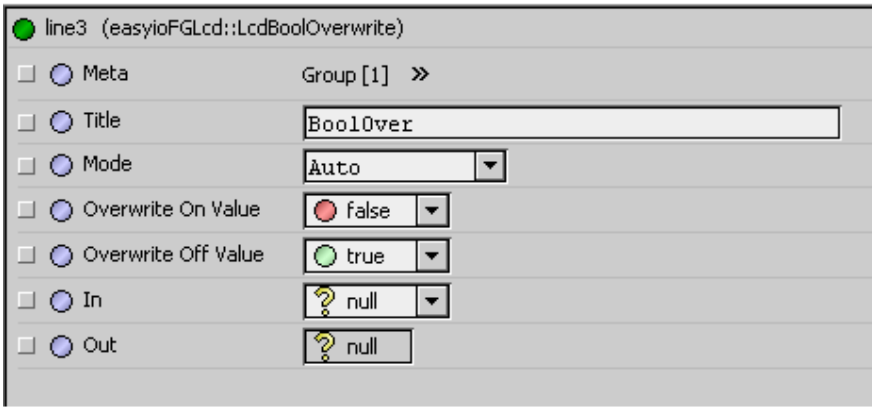
10.4 LcdBoolOverwrite

LcdBoolOverwerite is a Boolean object and child object that sits under the ***"page"*** object. It is used to control/overwrite Boolean point from the LCD display. It is a permanent overwrite until the overwrite is released.
It is a writable object from the LCD display.

It has 3 Selection ;

1. Auto , null value
2. Overwrite On
3. Overwrite Off

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **Mode**
  Selection of mode available. This selection is available in the LCD display.

  Auto , null value output. When this mode is selected , Out = In
  Overwrite On. When this mode is selected , Out = Overwrite On Value
  Overwrite Off. When this mode is selected , Out = Overwrite On Value

- **Overwrite On Value**
  User define output value when mode selection is Overwrite On.

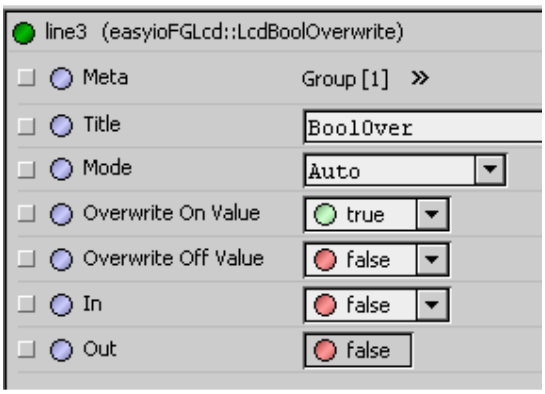
- **Overwrite Off Value**
  User define output value when mode selection is Overwrite Off.
- **In**
  Input constant value.

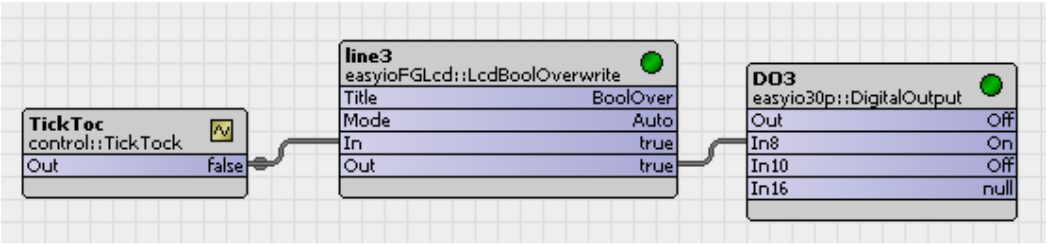
- **Out**
  Current Output Value. The control algorithm is a below.

If the input is set to true , overwrite value On set to true , overwrite value Off set to false ,
Mode = Auto
Output = Input

If the input is set to true , overwrite value On set to true , overwrite value Off set to false ,
Mode = Overwrite On
Output = Overwrite Value On = true

If the input is set to true , overwrite value On set to true , overwrite value Off set to false ,
Mode = Overwrite Off
Output = Overwrite Value Off = false





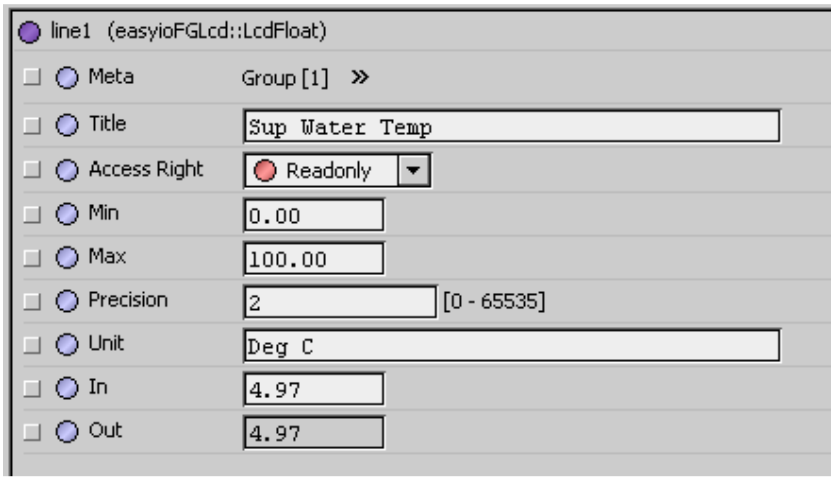
Example of using the LcdBoolOverwrite object

10.5 LcdFloat

LcdFloat is a Float object and child object that sits under the ***"page"*** object. It is used to display Float point in the LCD display.

It can be a read-only object or writable object.

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **Access Right**
  Selection between Read-only or Writable.
  If writable is selected, title will be in **Bold** is view from the LCD

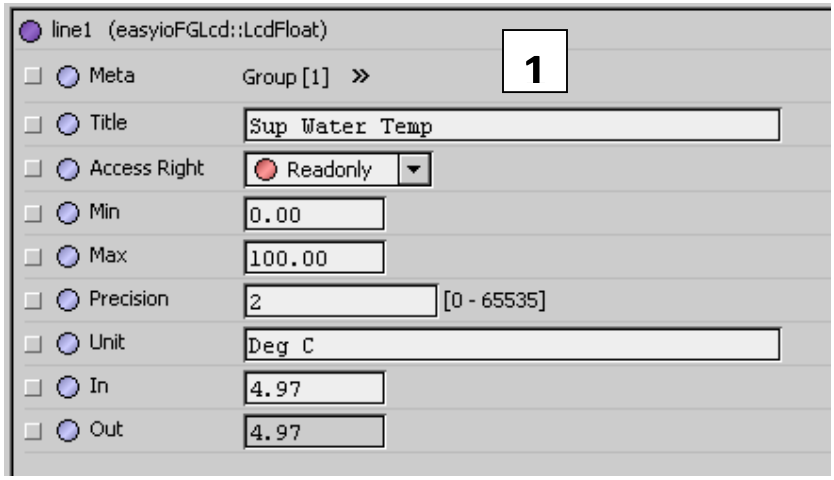
- **Min**
  Min value that the LCD Float can display

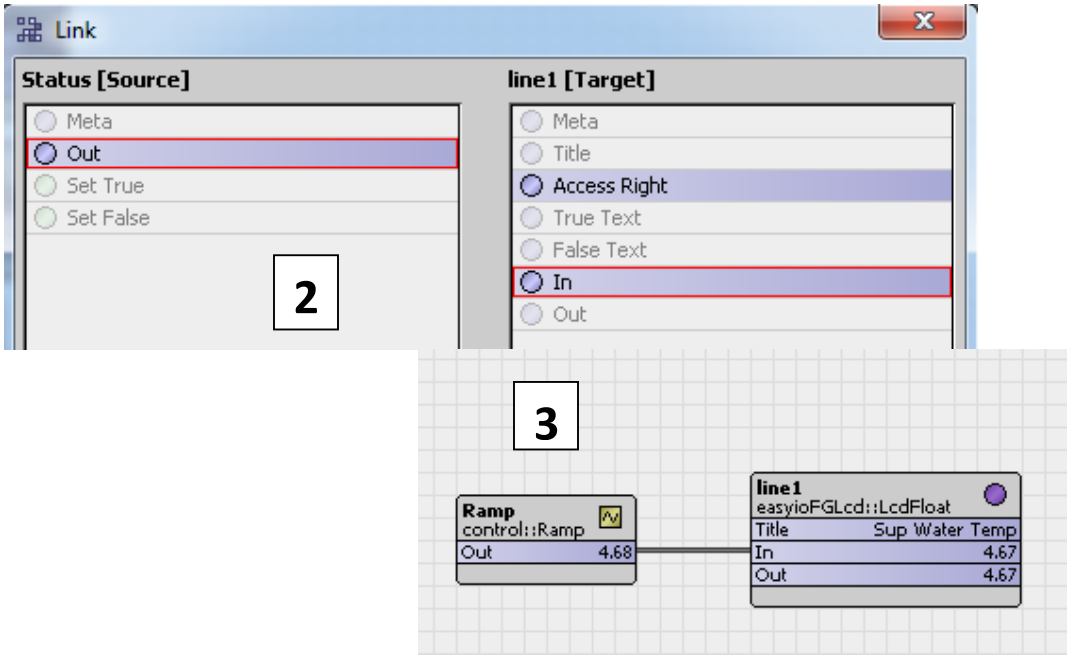
- **Max**
  Min value that the LCD Float can display

- **Precision**
  The number of precision value that the float value will display in the LCD screen

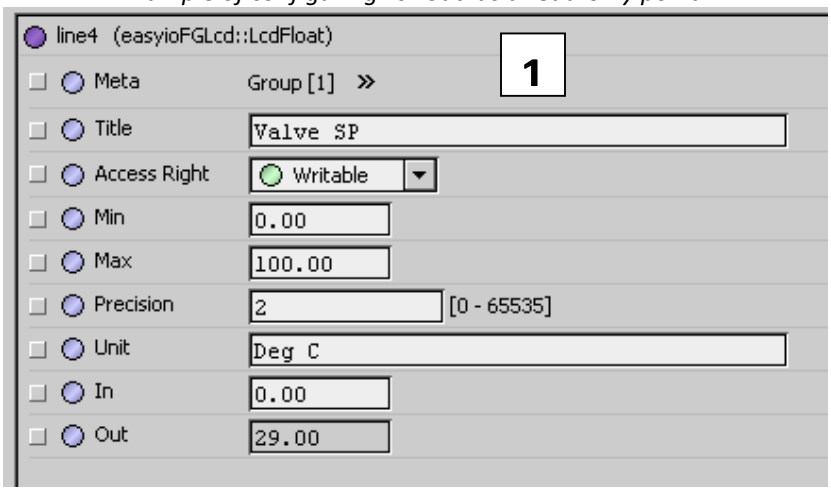
- **Unit**
  Engineering Unit that will be display next to the Float Value. This is a string value and max of ***15*** characters supported.
- **In**
  Input constant value

- **Out**
  Output value to the LCD display

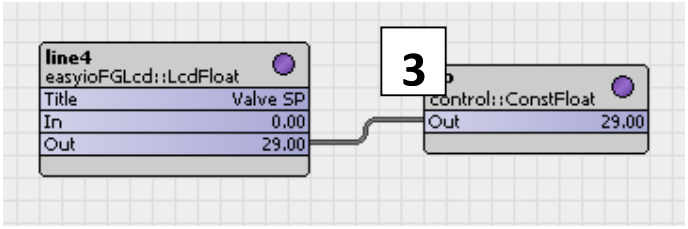




Example of configuring LcFloatl as a read-only point.



2



Example of configuring LcdFloat as a writable point. The LcdFloat object is writing to a Setpoint Object.

10.6 LcdFloatOneShot

LcdFloatOneShot is a Float object and child object that sits under the ***"page"*** object. It is used to control Float point from the LCD display for a define time period.
It is a writable object.

The property sheet of the object is shown below

- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **Retrigger Enable**
  This property can be set to enable is re-trigger is required.

- **Trigger period**
  Period of the trigger period in miliseconds.

- **Trigger Value**
  Output Value when object is triggered.

- **Precision**
  The number of precision value that the float value will display in the LCD screen

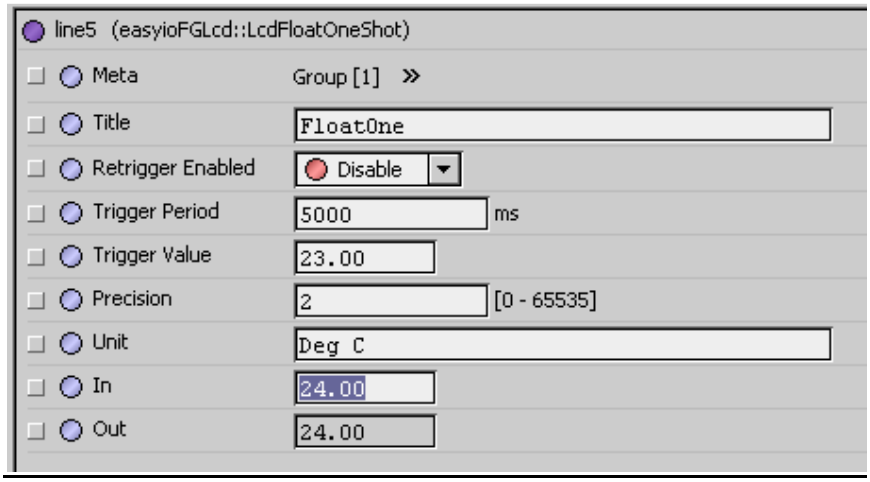
- **Unit**
  Engineering Unit that will be display next to the Float Value. This is a string value and max of ***15*** characters supported.

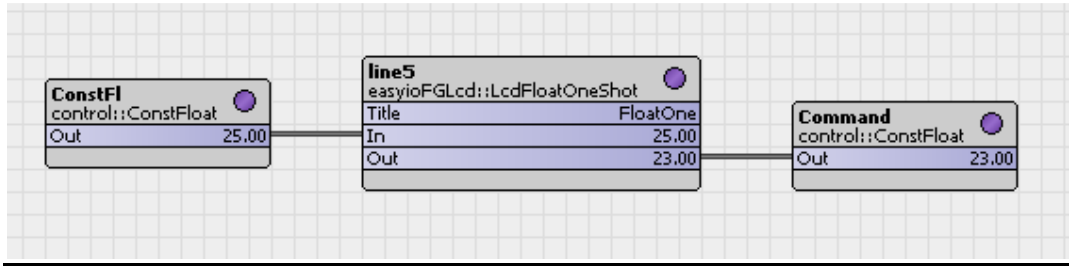
- **In**
  Input Constant value.

- **Out**
  Current Output Value. The control algorithm is a below.

  If the input is set to X , trigger value set toY , when a trigger fire

Output = Y according to the period time.





Example of LcdFloatOneShoot,

In = 24.00
Trigger value = 23.00
Out = Trigger Valuefor 5seconds when object is trigger from the LCD

Out = In after trigger period 5 second ends.

10.7 LcdFloatOverwrite

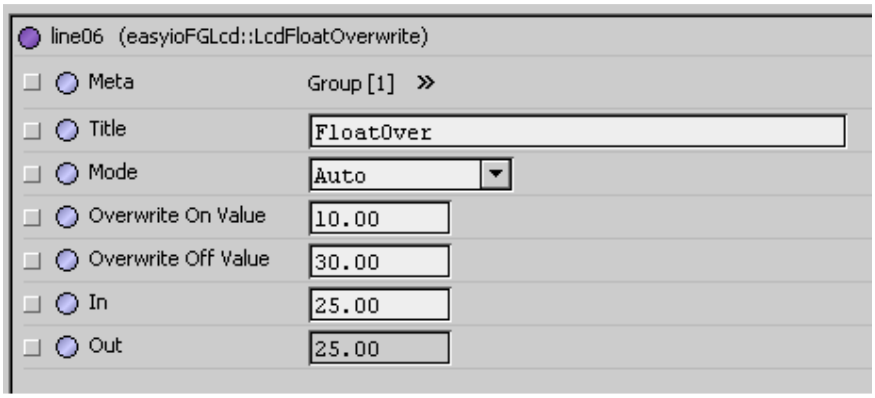
LcdFloatOverwerite is a Float object and child object that sits under the ***"page"*** object. It is used to control/overwrite Float point from the LCD display. It is a permanent overwrite until the overwrite is released.

It is a writable object from the LCD display.

It has 3 Selection ;

4. Auto , null value
5. Overwrite On
6. Overwrite Off

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **Mode**
  Selection of mode available. This selection is available in the LCD display.

  Auto , null value output. When this mode is selected , Out = In
  Overwrite On. When this mode is selected , Out = Overwrite On Value
  Overwrite Off. When this mode is selected , Out = Overwrite On Value

- **Overwrite On Value**
  User define output value when mode selection is Overwrite On.

- **Overwrite Off Value**
  User define output value when mode selection is Overwrite Off.

- **In**
  Input constant value.

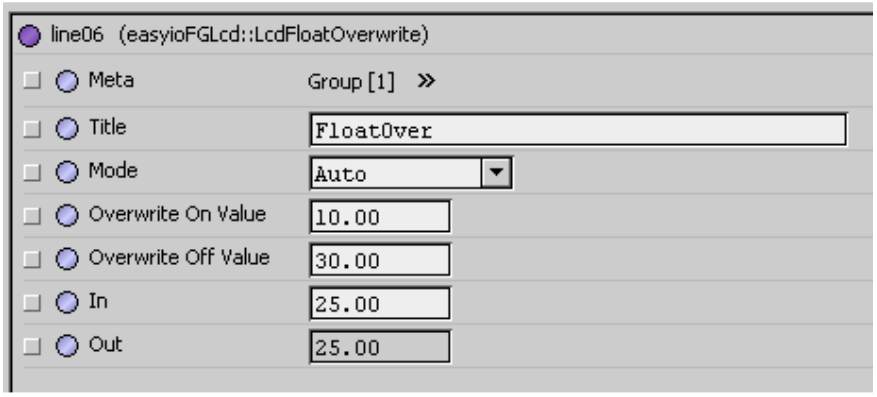
- **Out**

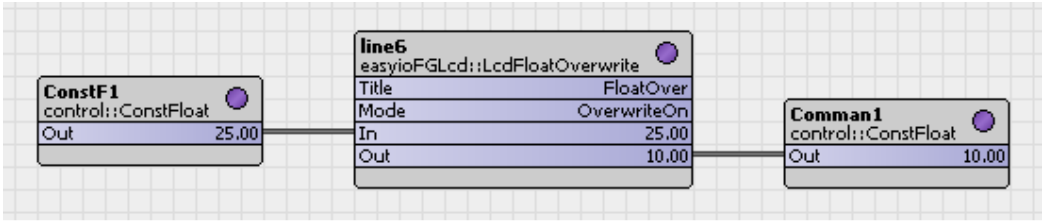
Current Output Value. The control algorithm is a below.

If the input is set to X , overwrite value On = Y , overwrite value Off = Z ,
Mode = Auto
Output = X

If the input is set to X , overwrite value On = Y , overwrite value Off = Z ,
Mode = Overwrite On
Output = Overwrite Value On = Y

If the input is set to X , overwrite value On = Y , overwrite value Off = Z ,
Mode = Overwrite Off
Output = Overwrite Value On = Z





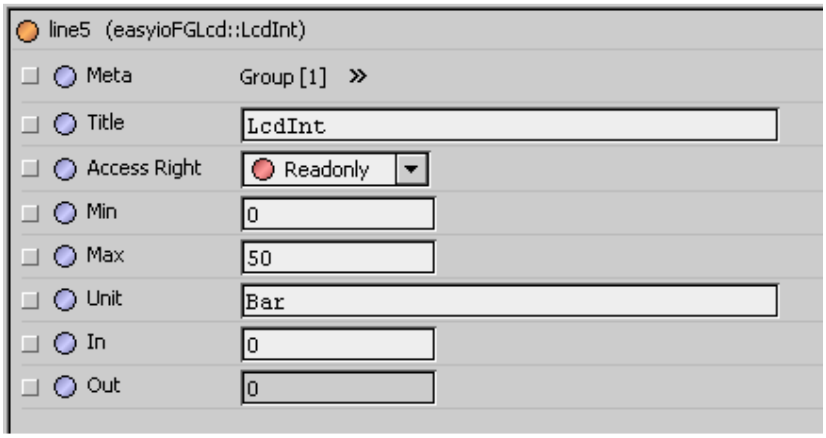
Example of using the LcFloatlOverwrite object

10.8 LcdInt

LcdInt is a Integer object and child object that sits under the ***"page"*** object. It is used to display Float point in the LCD display.

It can be a read-only object or writable object.

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **Access Right**
  Selection between Read-only or Writable.
  If writable is selected, title will be in **Bold** is view from the LCD

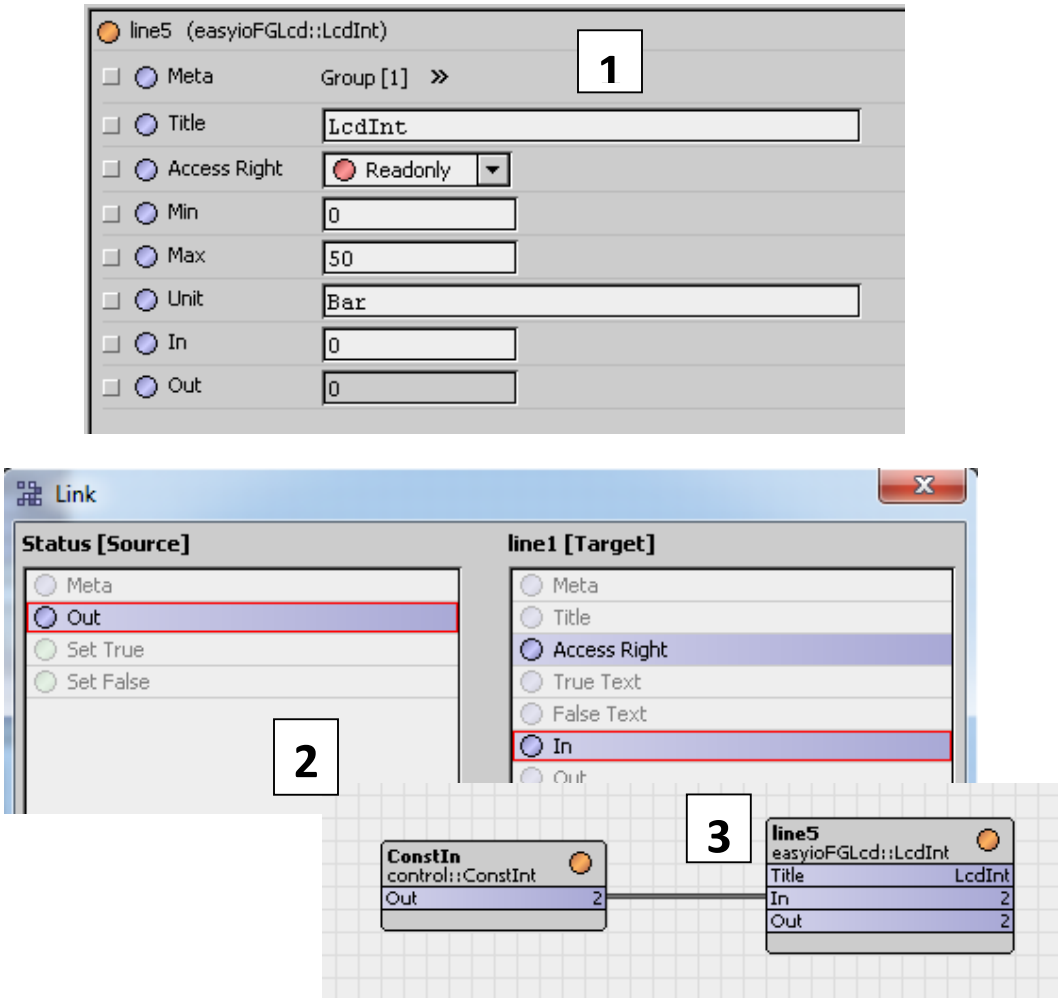
- **Min**
  Min value that the LCD Integer can display

- **Max**
  Min value that the LCD Integer can display

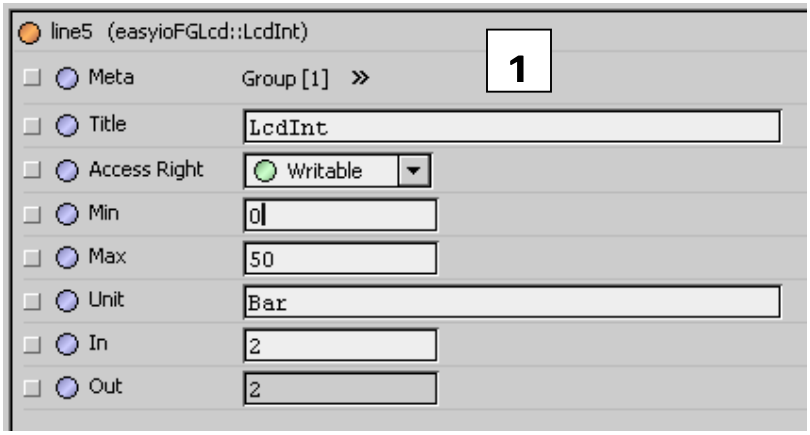
- **Precision**
  The number of precision value that the float value will display in the LCD screen

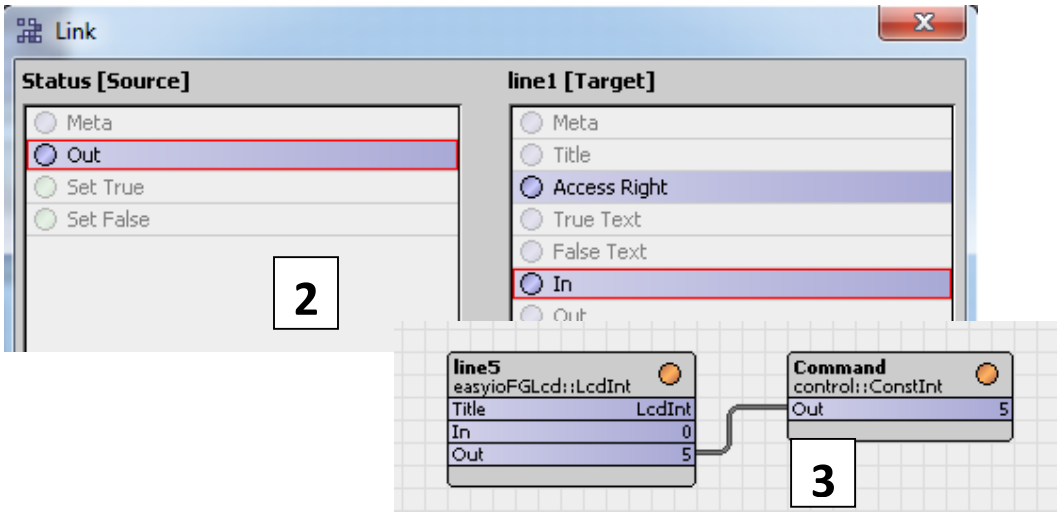
- **Unit**
  Engineering Unit that will be display next to the Float Value. This is a string value and max of ***15*** characters supported.
- **In**
  Input constant value

- **Out**
  Output value to the LCD display



Example of configuring LcFloatl as a read-only point.





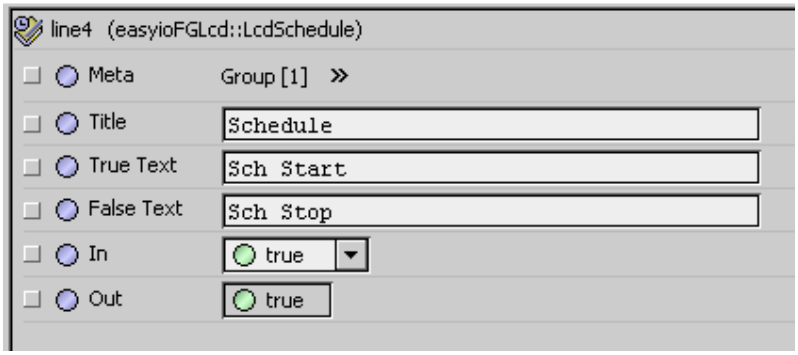
Example of configuring LcdInt as a writable point. The LcdInt object is writing to a object.

10.9 LcdSchedule

LcdSchedule is a Integer object and child object that sits under the ***"page"*** object. It is used to display easyIO schedule object in the LCD display.

This object will replicate the easyIO schedule in the sedona apps and display it in the LCD display. User can edit the schedule from the LCD display with the same format as easyIO schedule object. Please refer to easyIO sedona kits , easyioSchedule for schedule editing format.

The property sheet of the object is shown below



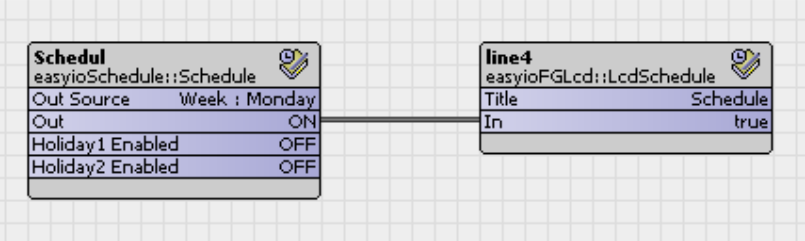
- **Title**
  Title for the point to be shown in the LCD screen.
  Max 15 characters.

- **True text**
  Text that will appear in the LCD display next to the title

- **True text**
  Text that will appear in the LCD display next to the title

- **In**
  Input constant value.
  This slot has to be link from a easyioSchedule kit object.

- **Out**
  Output value to other objects to be control.



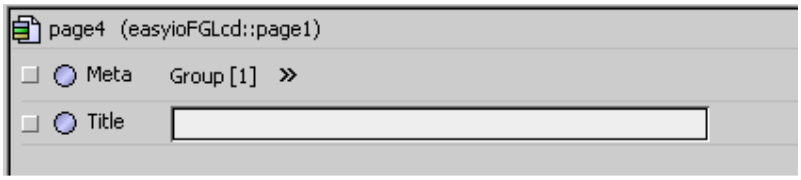
Example of using the LcdSchedule object in the sedona apps.

10.10 Page

Page is a child object for easyioLcdService. It must be a child of the LCD service. This object will determine the page that will display in the LCD display.

Max number of pages allowed per controller at the moment is 99 pages.

The property sheet of the object is shown below



- **Title**
  Title for the point to be shown in the LCD screen page view. This title will appear on top of the page.
  Max 15 characters are allowed.

  Title is not used for ***hpage***.

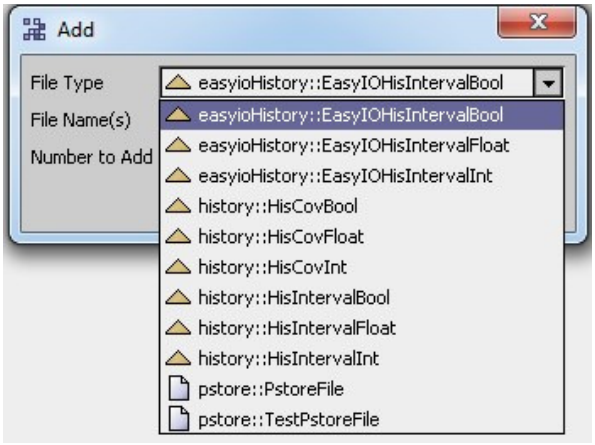
12 EasyioHistory

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
11	easyioHistory	1.0.45	Easyio 1.0.43. or higher Tridium Pstore Tridium History	EasyIOHistoryIntervalBool EasyIOHistoryIntervalFlaot EasyIOHistoryIntervalInt

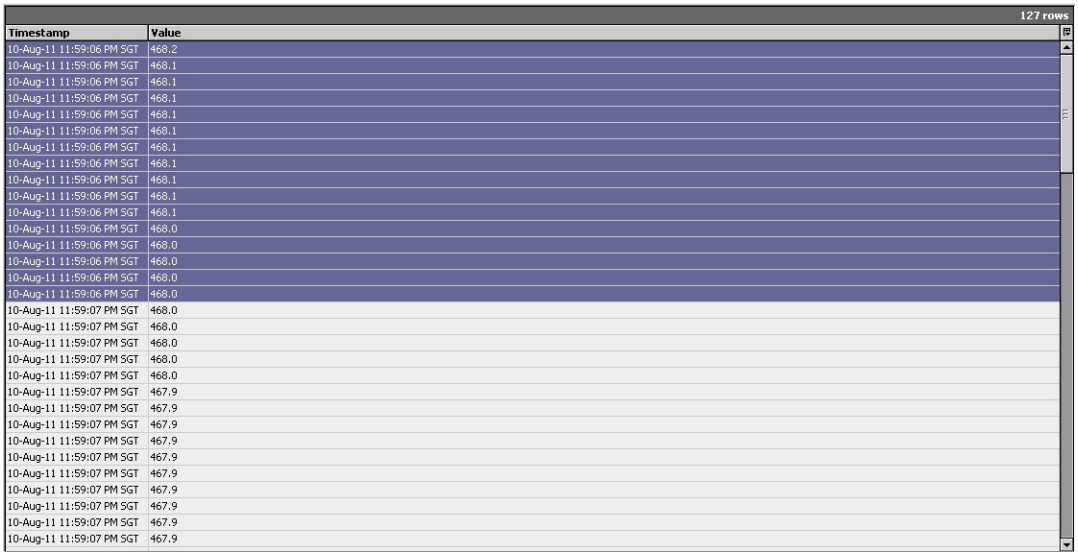
This kit contains 3 objects. All the objects extend the Tridium history kit for Interval history logs.

These object eliminates the daily transition error where it will create 20 lines per second where by the interval is set to minutely.

Objects available in the PStore Service property sheet.



Below is some example of the tridium Interval history error. Every second has 20 lines.

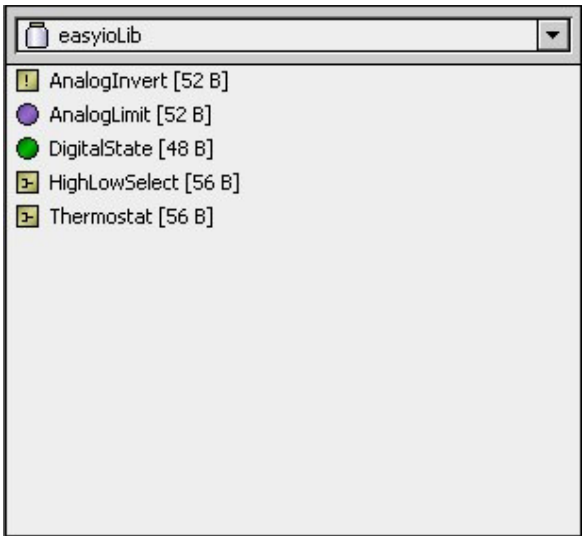


13 EasyioLib

Number	EasyIO Sedona Kit	Current Version	Dependencies	Remarks
12	easyioLib	1.0.43	Easyio 1.0.43. or higher	AnalogInvert AnalogLimit DigitalState HighLowSelect Thermostat

This kit contains 5 objects. All the objects can be used for the controller logic programming.

To use these objects, simply just drag and drop into the wire sheet.

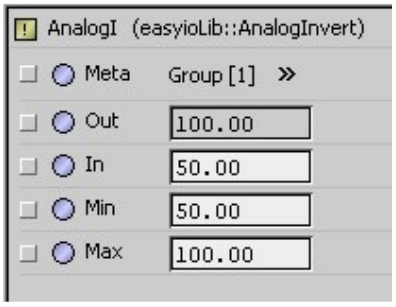


13.1 *AnalogFilter*

AnalogInvert component invert the input based on scale factor.

```
if In < Min then Out = Max
else if In > Max then Out = Min
else Out = Max + Min - In
```

The property sheet of the object is shown below



- **Out**
  Current output value. Readonly
- **In**
  Current input value.

- **Min**
  Minimum input value, scale low factor.

- **Max**
  Maximum input value, scale high factor.

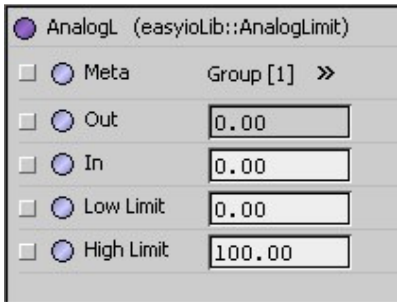
Example of AnalogInvert object used

13.2 *AnalogLimit*

AnalogLimit component restricts an analog value to a specific range. When the input value is within the limit range, it will be passed direct to the output.

```
if In < LowLimit then Out = LowLimit
else if In > HighLimit then Out = HighLimit
else Out = In
```

The property sheet of the object is shown below

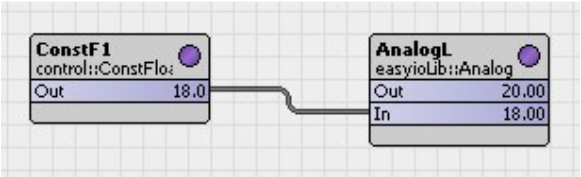


- **Out**
  Current output value. Readonly

- **In**
  Current input value.

- **Low Limit**
  Lowest allowed value.

- **High Limit**
  Maximum allowed value.

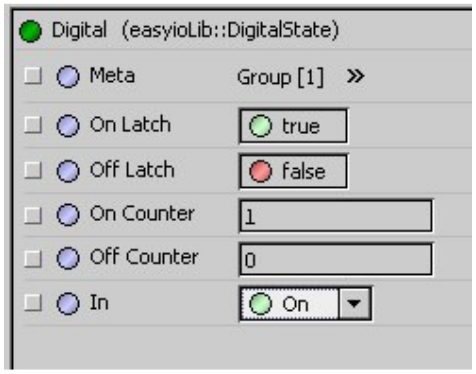


Example of AnalogLimit used to limit the value of the Input. Min Limit is set to 20 and Max Value is set to 30. This application can be used to limit the temperature setpoint.

13.3 *DigitalState*

DigitalState component monitors the digital state transition from On to Off state and Off to On state (On/Off) latch and monitors the digital state On to Off transition and Off to On transition count.

The property sheet of the object is shown below



- **On Latch**
  OnCounter provides a means to count OFF to ON digital transition on the Out state. This output increments by one on each In state OFF to ON transition. Readonly

- **Off Latch**
  OffLatch is the ON to OFF transition capture at the In state. This output remains in ON state following the first ON to OFF transition unless reset by ResetOffLatch action. Readonly
  true = At least one ON to OFF transition
  false = No ON to OFF transition occurred or reset by ResetOffLatch action.

- **On Counter**
  OnLatch is the OFF to ON transition capture at the In state. This output remains in ON state following the first OFF to ON transition unless reset by ResetOnLatch action. Readonly
  true = At least one OFF to ON transition
  false = No OFF to ON transition occurred or reset by ResetOnLatch action.

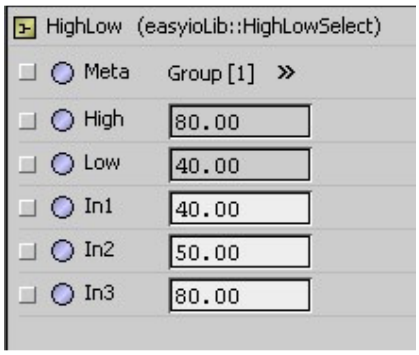
- **Off Counter**
  OffCounter provides a means to count ON to OFF digital transition on the Out state. This output increments by one on each In state ON to OFF transition. Readonly

- **In**
  Current input state.

13.4 *HighLowSelect*

HighLowSelect component output highest and lowest value out of 3 input values.

The property sheet of the object is shown below



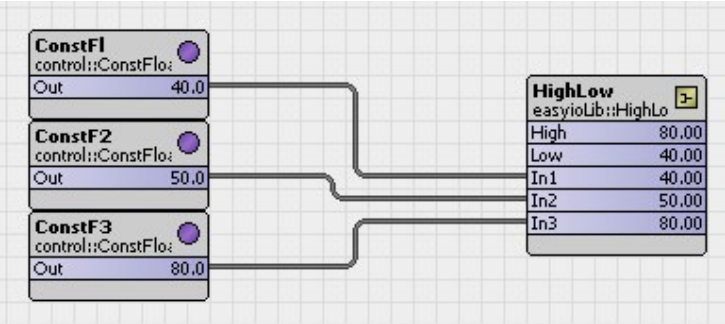
- **High**
  The highest value of all inputs. Readonly

- **Low**
  The lowest value of all inputs. Readonly

- **In1**
  Input value 1.

- **In2**
  Input value 2.

- **In3**
  Input value 3.

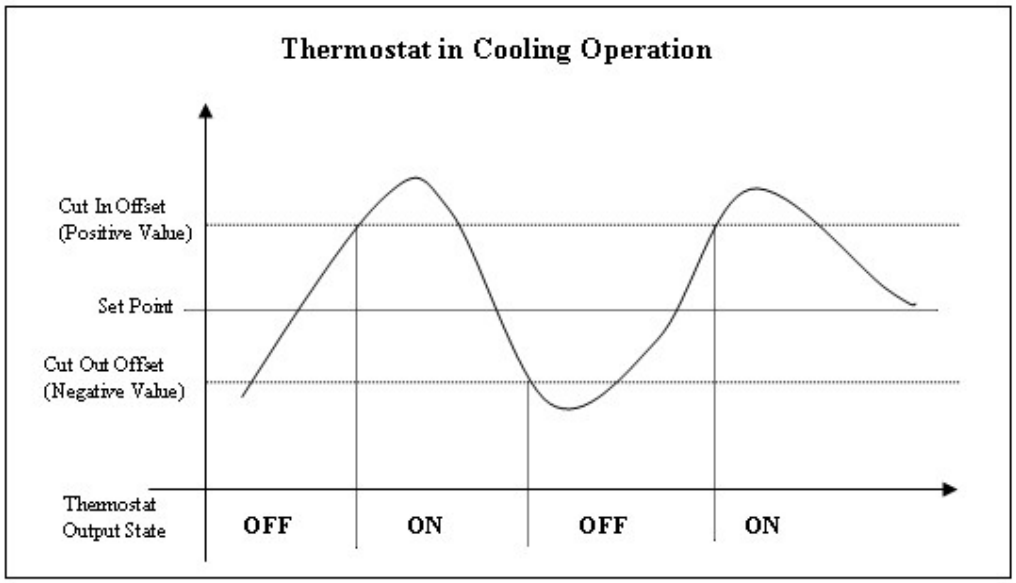


Example of a highlow object with 3 inputs

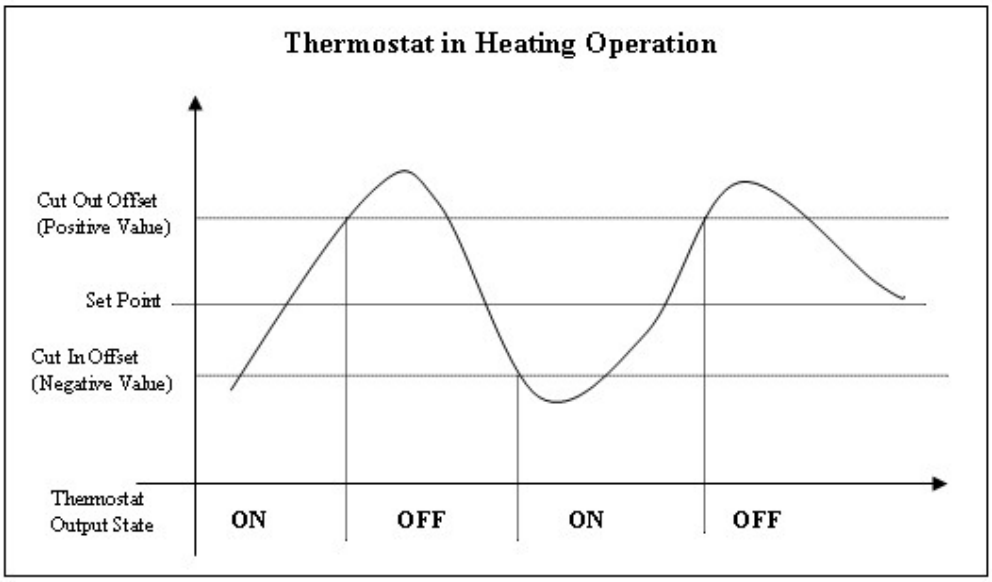
13.5 *AnalogFilter*

Thermostat component provides the output control based on the input (process) and the set point value.

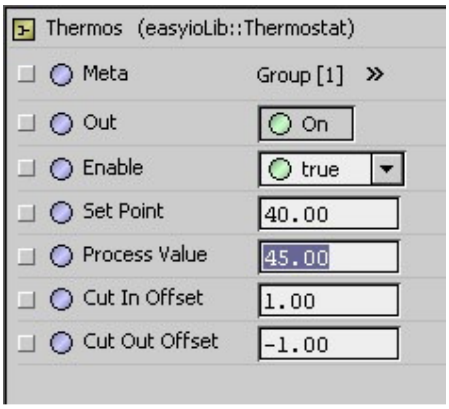
Thermostat in cooling operation:



Thermostat in heating operation:



The property sheet of the object is shown below



- **Out**
  Output state, ON or OFF as the comparison result of ProcessValue and SetPoint. Readonly

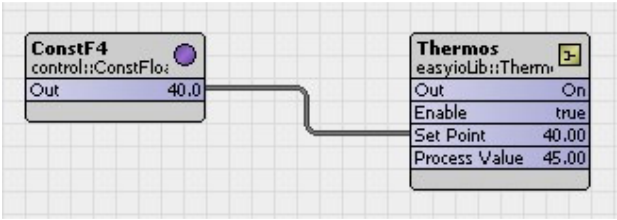
- **Enable**
  Enable the thermostat function.

- **Set Point**
  Desired/target value.

- **Process Value**
  Thermostat input value. Thermostat function block compares the SetPoint and the ProcessValue to determine the output state.

- **Cut In Offset**
  Defines the differential value between ProcessValue and SetPoint to determine the Thermostat output on state. A positive CutInOffset value means greater than SetPoint, and a negative CutInOffset value means lower than SetPoint during comparison. For cooling control, uses positive value and negative value for heating control.

- **Cut Out Offset**
  Defines the differential value between ProcessValue and SetPoint to determine the Thermostat output off state. A positive CutOutOffset value means greater than SetPoint, and a negative CutOutOffset value means lower than SetPoint during comparison. For cooling control, uses negative value and positive value for heating control.



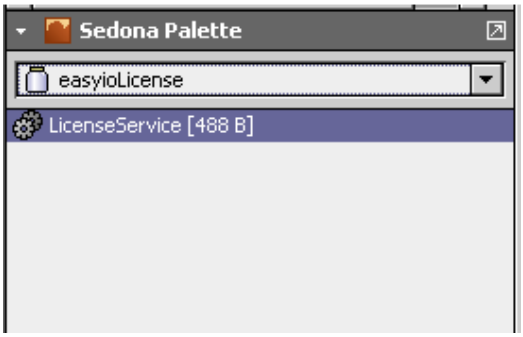
Example of thermostat object use

14 EasyioLicense

Number	EasyIO Sedona Kit	Current Version	Dependencies	Remarks
13	easyioLicense	1.0.45	Easyio 1.0.43.10 or higher	LicenseService

easyioLicense kit function as the centralize management for the license of all common kits.
easyioLicense kit contains 1 objects

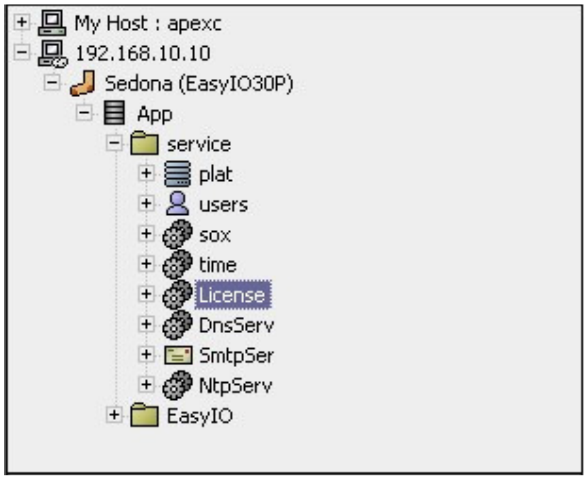
To use these objects just drag and drop into the service folder wire sheet.



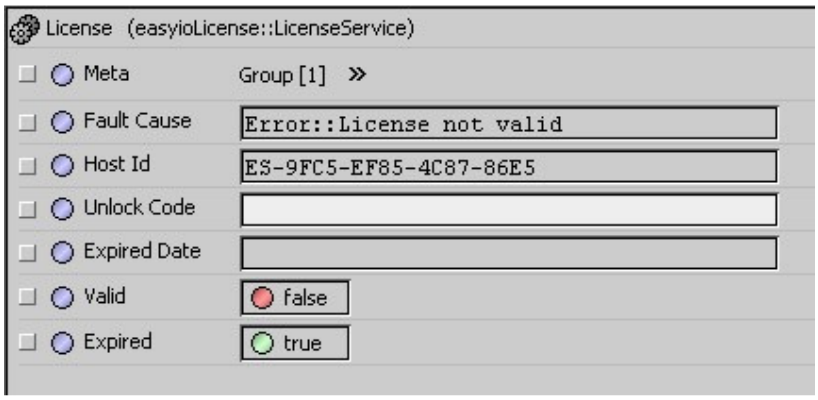
14.1 *LicenseService*

LicenseService is a service to manage the license for all common kits.

****Note: LicenseService must be drop inside Service folder (Sedona -> App -> Service).**



The property sheet of the object is shown below



- **Fault Cause**
  To show cause of the error, when there was LicenseService failure.

- **Host Id**
  Device Host ID

- **Unlock code**
  A series of alphabets or numbers, as the license keys to unlock all the common kits.

- **Expired Date**
  Expiry date of the unlock code (license).

- **Valid**
  To show the status whether the license/unlock code is valid. True for valid and false for invalid license.

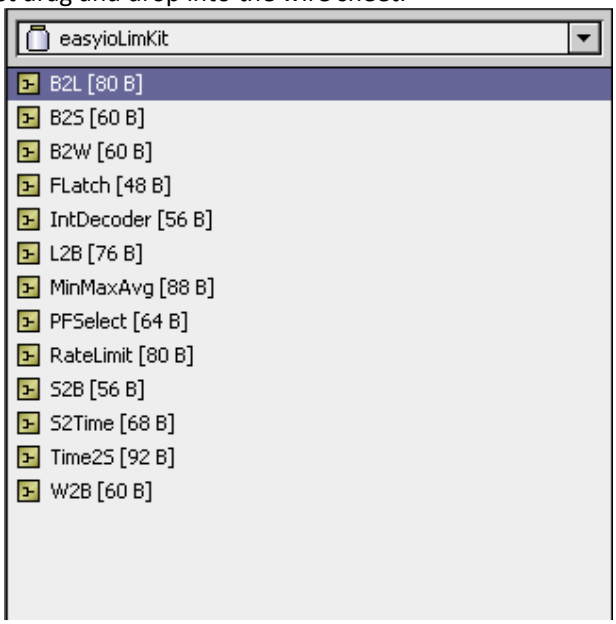
- **Expired**
  To show the status whether the license/unlock code is expired. An expired license will show true.

15 EasyioLimkit

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
15	EasyioLimkit	1.0.45.3	Easyio 1.0.43.10 or higher Firmware 0.5.00 and later	B2L B2S B2W FLatch IntDecoder L2B MinMaxAvg PFSelect RateLimit S2B S2Time Time2S W2B

This kit contains 13 objects. These objects are fundamentally for conversions.

To use these objects just drag and drop into the wire sheet.



15.1 *B2L*

B2L or Bit to Long conversion object. The output data type is “long”

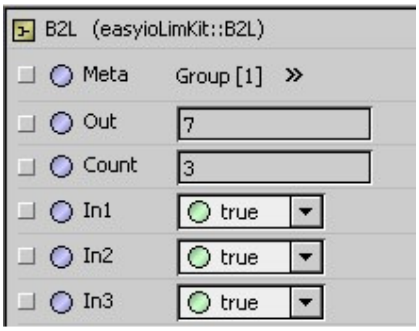
The property sheet of the object is shown below

- **Out**
  Output of the conversion base on the binary.
  The output data type is “long”

- **Count**
  This slot shows the total number of bit count. It will count total number of bit between bit1 to bit32 which the value is “true”.

- **In1, In2, In3 ……. In32**
  There are total 32 input for the object to calculate the binary format.

  In1 = LSB (Least Significant Bit)
  In32 = MSB (Most Significant Bit)



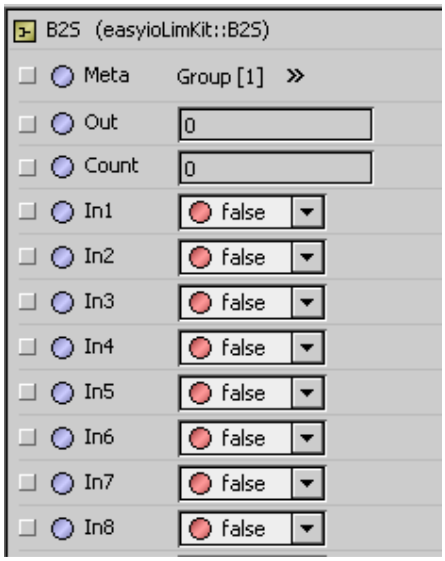
Example of B2L object use

$$Out = In1 + In2 + In3$$
$$Out = 2^0 + 2^1 + 2^2$$
$$Out = 7$$

15.2 *B2S*

B2S or Bit to Short conversion object. The output data type is “short”

The property sheet of the object is shown below

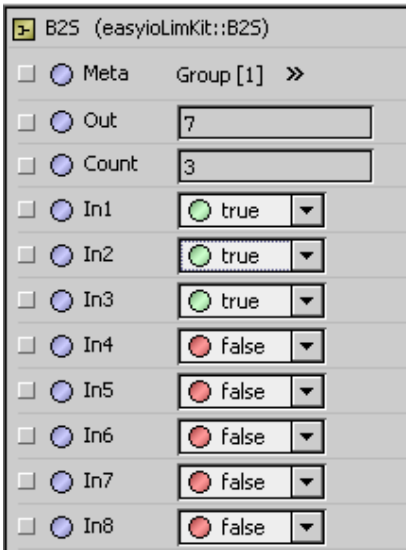


- **Out**
  Output of the conversion base on the binary
  The output data type is “word”

- **Count**
  This slot shows the total number of bit count. It will count total number of bit between bit1 to bit16 which the value is “true”.
- **In1, In2, In3 ……. In16**

There are total 16 input for the object to calculate the binary format.

In1 = LSB (Least Significant Bit)
In32 = MSB (Most Significant Bit)



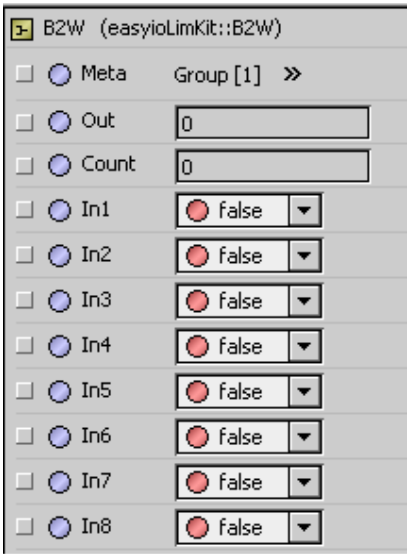
Example of B2S object use

$$Out = In1 + In2 + In3$$
$$Out = 2^0 + 2^1 + 2^2$$
$$Out = 7$$

15.3 *B2W*

B2W or Bit to Word conversion object. The output data type is “word”

The property sheet of the object is shown below

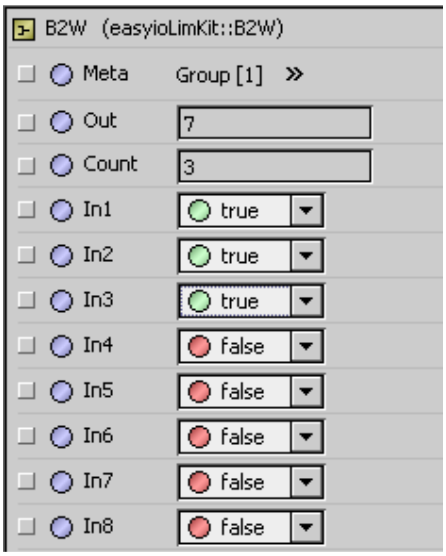


- **Out**
  Output of the conversion base on the binary
  The output data type is “word”

- **Count**
  This slot shows the total number of bit count. It will count total number of bit between bit1 to bit16 which the value is “true”.

- **In1, In2, In3 ……. In16**
  There are total 16 input for the object to calculate the binary format.

  In1 = LSB (Least Significant Bit)
  In32 = MSB (Most Significant Bit)



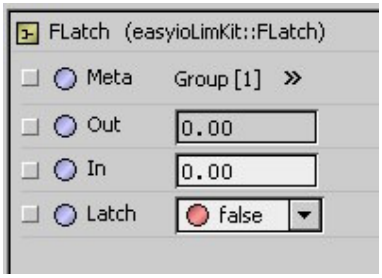
Example of B2W object use

$$Out = In1 + In2 + In3$$
$$Out = 2^0 + 2^1 + 2^2$$
$$Out = 7$$

15.4 *FLAtch*

FLatch is a latching object where it can latch a float input value.

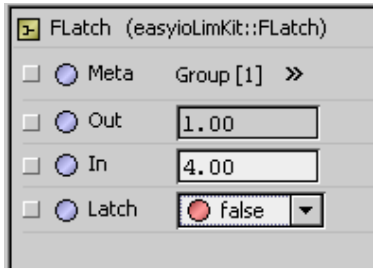
The property sheet of the object is shown below



- **Out**
  Output value when the latch is triggered

- **In**
  Input variable for the latching process

- **Latch**
  This slot is link to a Boolean to trigger the latch. The output value will be latch until the next latch slot true.

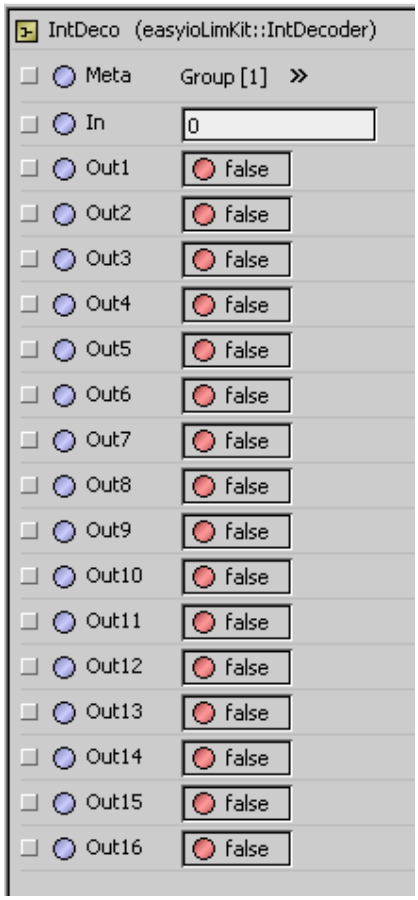


Example of FLatch object use

15.5 *IntDecoder*

IntDecoder is an object where it converts an integer value and triggers a Boolean according to the input value.

The property sheet of the object is shown below



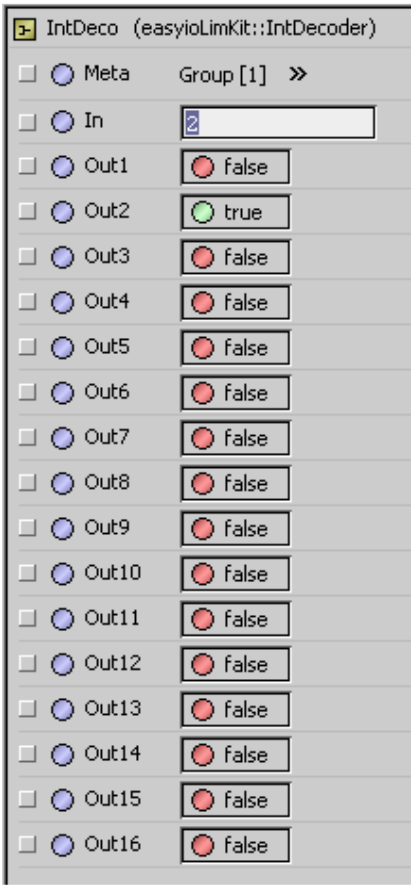
- **In**
  Input value. An integer format value.

- **Out**
  16 boolean outputs.

  If In = 0
  All Out = false

  If In = 1
  Out1 = true
  Out 2, Out3…Out16 = false

  If In = 4
  Out 1, Out2, Out3, Out4 = true
  Out5, Out6, Out7….Out16 = false

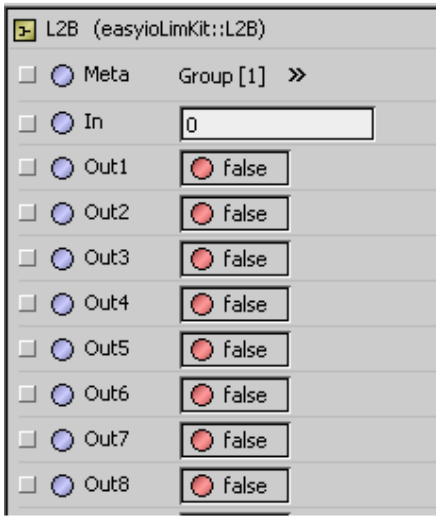


Example of IntDecoder object use

15.6 *L2B*

L2B or Long to Bit conversion object. The output data type is Boolean. 32 outputs available.

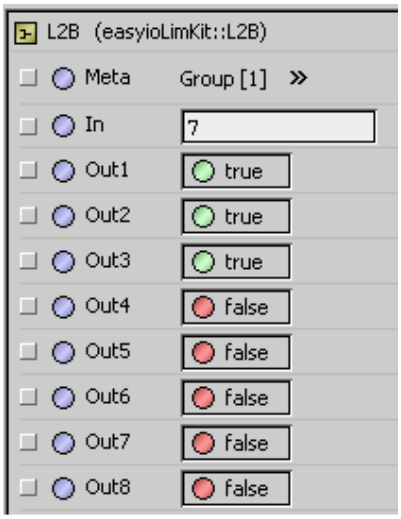
The property sheet of the object is shown below



- **In**
  Input of the conversion base on the long data type input.
  The output data type is “boolean”

- **Out1, Out2, Out3 ……. Out32**
  There are total 32 outputs for the object which supports up to a 32bit value.

  Out1 = LSB (Least Significant Bit)
  Out32 = MSB (Most Significant Bit)



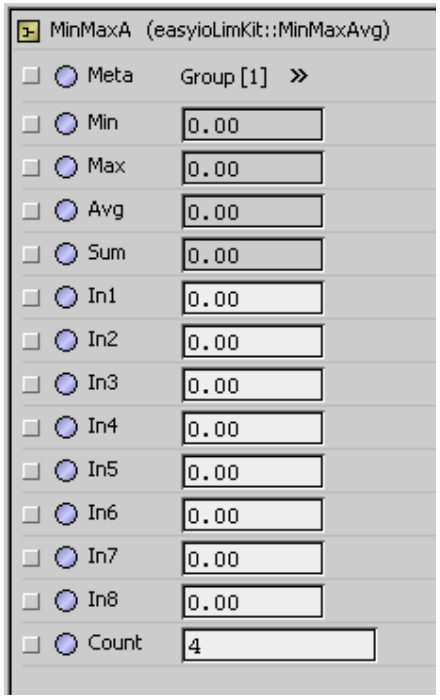
Example of L2B object use

In = 7
Out1 = true
Out2 = true
Out3 = true
Out4 to Out 32 = false

15.7 *MinMaxAvg*

MinMaxAvg is an object that will calculate the Min, Max, Summation and Average of the inputs. It supports up to 8 inputs.

The property sheet of the object is shown below



- **Min**
  Minimum value base on 8 inputs.

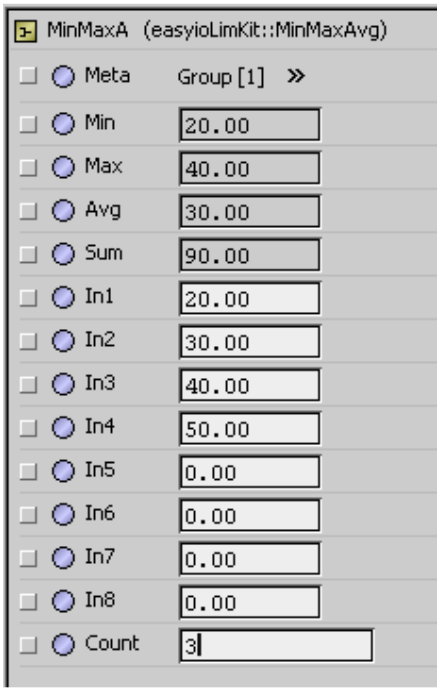
- **Max**
  Maximum value base on 8 inputs.

- **Avg**
  Calculated average value base on Count specify in the count property.

- **Sum**
  Calculated summation value base on Count specify in the count property.

- **In1, In2…..In8**
  Input variable. Up to 8 inputs can be use.

- **Count**
  Property to enable total number of inputs that to be use for calculation.

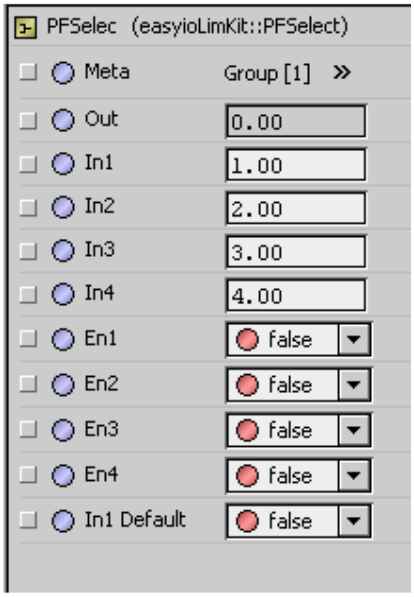


Example of MinMaxAvg object use
In this example, it only computes 3 inputs as the count is specify to 3.

15.8 *PFloatSelect*

PFSelect is Priority Select. It only supports float value. There are 4 inputs floats value and 4 input Enable.

The property sheet of the object is shown below



- **Out**
  Output value as per Enable selection

- **In1, In2, In3 and In4**
  User define input float value.

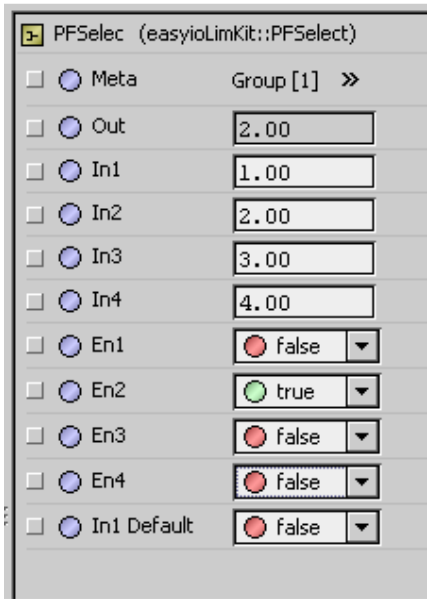
- **En1, En2, En3 and En4**
  Value enable for respective inputs.

  En1 = true , Out = In1
  En2 = true , Out = In2
  En3 = true , Out = In3
  En4 = true , Out = In4

  Only 1 enable can be true in any condition. If 2 Enable are true, output will be the higher priority value that enable.

- **In1 Default**
  If set to true, default value is In1 if all enable are false. This is to eliminate output to be zero in case the entire enable are false.

  If set to false, out value will be zero if all enable are false.

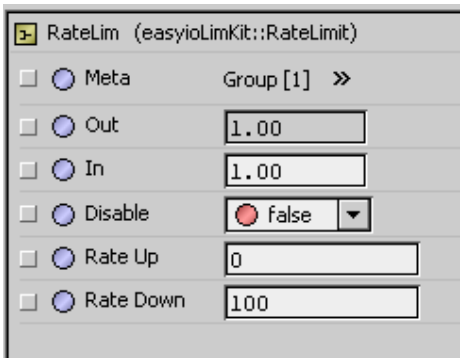


Example of PFSelect object use

15.9 *RateLimit*

RateLimit is an object to reduce the update rate for a float value. In some cases, where a fast value change will caused a fast ramp in the equipment. This object can slow down the value change by ramping the value slowly.

The property sheet of the object is shown below



- **Out**
  Output value after the a rate up or rate down process

- **In**
  Input value for the ramp limit object.

- **Disable**
  To disable the rate limiter.

- **Rate Up**
  Rate of increasing the value if there is any change in the In slot.
  Zero = disable.
  Higher the value the slower the change of the Out value will be.

- **Rate Down**
  Rate of decreasing the value if there is any change in the In slot.
  Zero = disable
  Higher the value the slower the change of the Out value will be.

The algorithm of the rate up and rate down is as below.

If the Diff > 0.0
*Out =(Diff) / (rateUp * 1sec)*

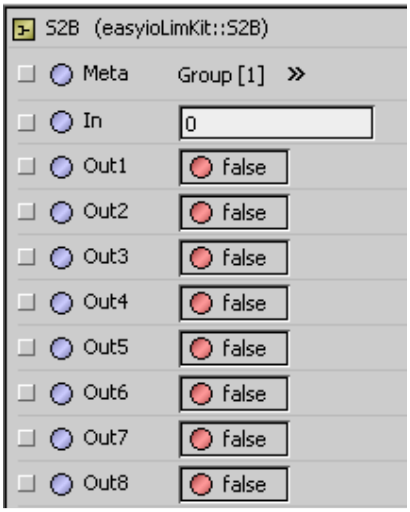
If the Diff < 0.0
*Out =(Diff) / (rateUp * 1sec)*

Where Diff = Input Last Value – Input Current Value

15.10 *S2B*

S2B or Short to Bit conversion object. The output data type is Boolean. 16 outputs available.

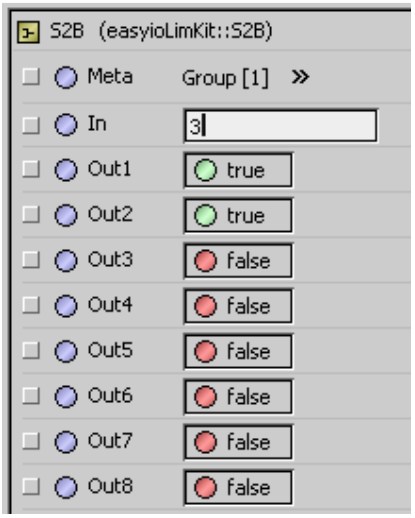
The property sheet of the object is shown below



- **In**
  Input of the conversion base on the short data type input.
  The output data type is “boolean”

- **Out1, Out2, Out3 ……. Out16**
  There are total 16 outputs for the object which supports up to a 16bit value.

  Out1 = LSB (Least Significant Bit)
  Out32 = MSB (Most Significant Bit)

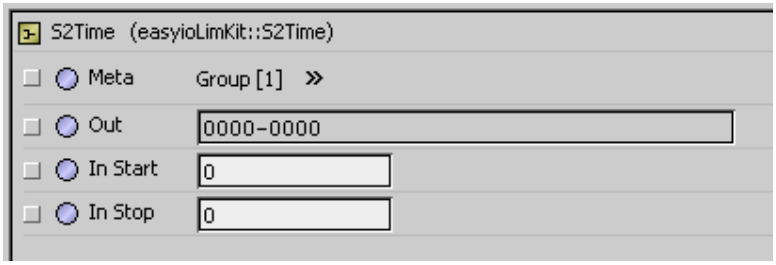


Example of S2B object use

In = 3
Out1 = true
Out2 = true
Out3 to Out16 = false

15.11 *S2Time*

S2Time or Short to Time conversion object. The output of this object is a time range in 24 hours format. This output can be used for the easyIO Boolean schedule time input string.
The property sheet of the object is shown below



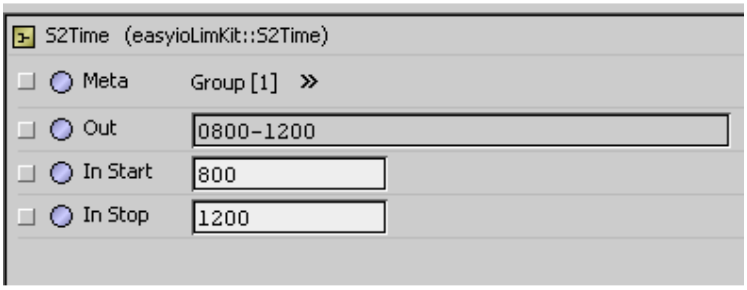
- **Out**
  Output of the conversion is time range, 24 hours format.
  The output data type is a string data type

- **In Start**
  The start time. Format is in 24 hours format.

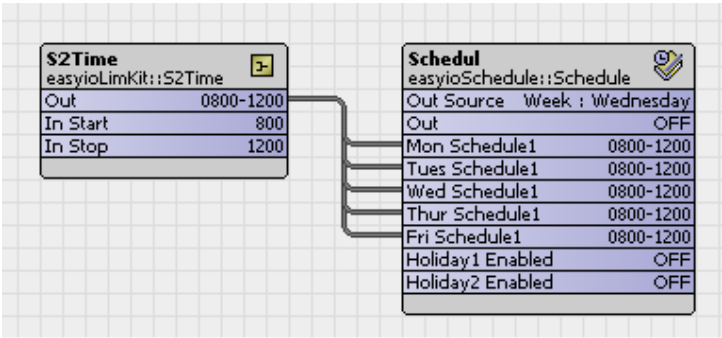
  0000 = 12.00AM
  2399 = 11.59PM

- **In Stop**
  The stop time. Format is in 24 hours format.

  0000 = 12.00AM
  2399 = 11.59PM

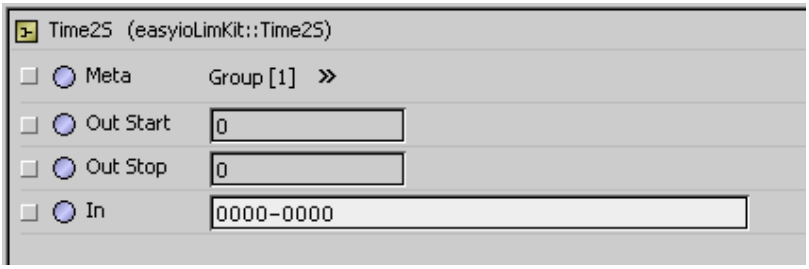


Example of S2Time object use.
The time range period is 8.00AM to 12.00PM is the on time.
User then can link this to the easyIO schedule.



15.12 *Time2S*

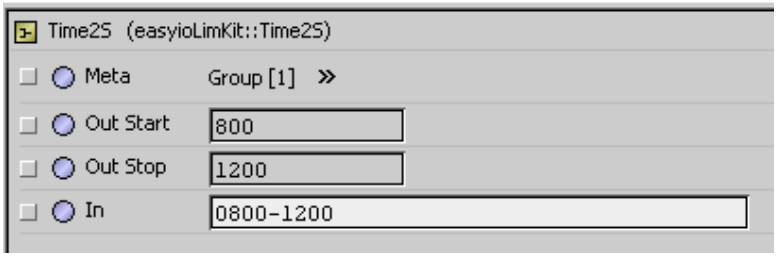
Time2S or Time to Short conversion object. The output of this object is a absolute time in 24 hours format.
The property sheet of the object is shown below



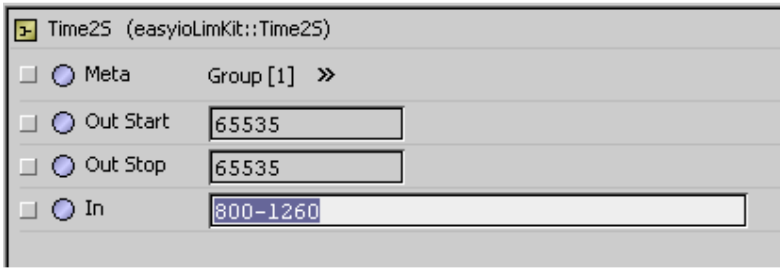
- **Out Start**
  Output of the conversion is absolute time, 24 hours format.
  The output data type is a short data type

- **Out Stop**
  Output of the conversion is absolute time, 24 hours format.
  The output data type is a short data type

- **In**
  This is the input time range of a string data type.
  The time format is in 24 hours format.
  Every segment must have 4 digits.
  Example 8.00AM = 0800
  12.00PM = 1200
  Failure to comply to the above , may cause wrong conversion.



Example of S2Time object use.

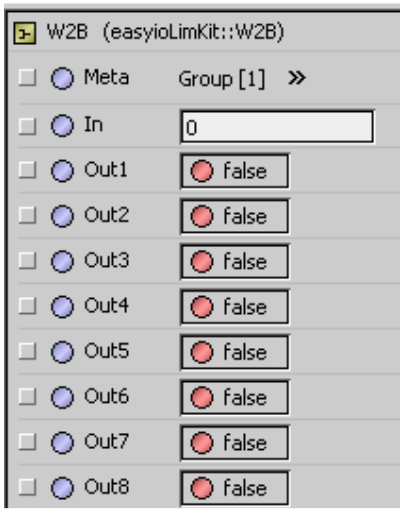


Example of S2Time object with incorrect time settings.

15.13 *W2B*

W2B or Word to Bit conversion object. The output data type is Boolean. 16 outputs available.

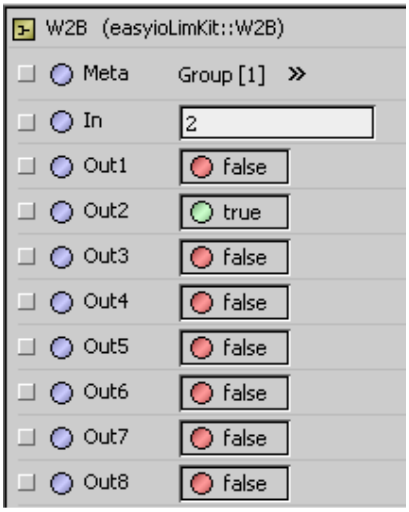
The property sheet of the object is shown below



- **In**
  Input of the conversion base on the integer data type input.
  The output data type is “boolean”

- **Out1, Out2, Out3 ……. Out16**
  There are total 16 outputs for the object which supports up to a 32bit value.

  Out1 = LSB (Least Significant Bit)
  Out32 = MSB (Most Significant Bit)



Example of W2B object use

In =2
Out2 = true
Out1 = false
Out3 to Out16 = false

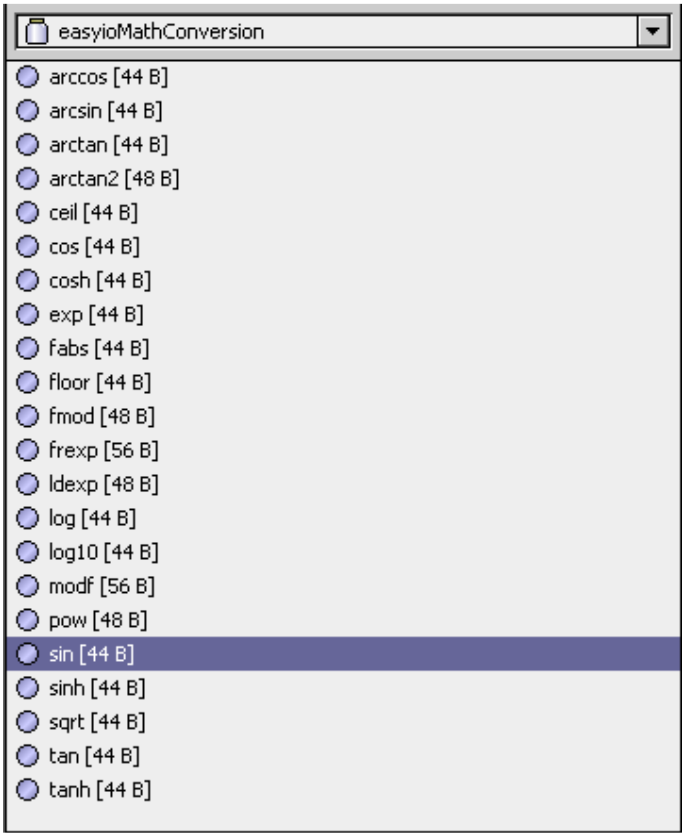
16 EasyioMathConversion

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
15	EasyioMathConversion	1.0.45.21	Easyio 1.0.43.10 or higher Firmware 0.5.00 and later	Arcos Arcsin Arctan Arctan2 Ceil Cos Cosh Exp Fabs Floor Fmod Frexp Ldexp Log Log10 Modf Pow Sin Sinh Sqrt Tan Tanh

This kit contains 22 objects. All the objects are to be used for mathematics calculations.

All the objects are derive from the C++ reference library. For details explainaton , refer to http://www.cplusplus.com/reference/clibrary/cmath/sin/

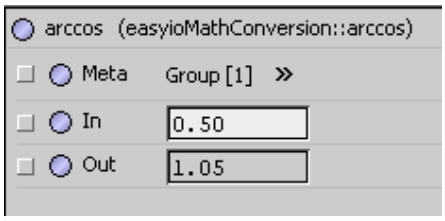
To use these objects just drag and drop into the wire sheet.



16.1 *ArcCosine*

Arccos or ArcCosine object.

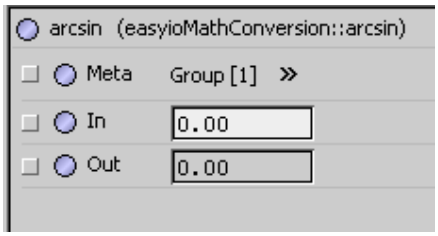
The property sheet of the object is shown below



16.2 *ArcSine*

Arcsin or ArcSine object

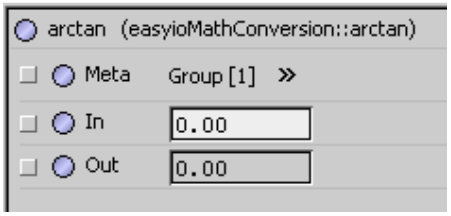
The property sheet of the object is shown below



16.3 *Arc Tangent*

Arctan or ArcTangent object

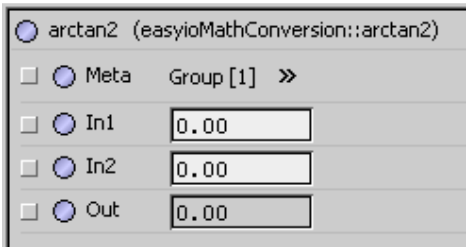
The property sheet of the object is shown below



16.4 *Arc Tangent 2*

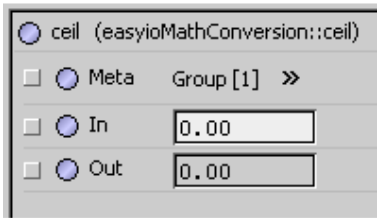
Arctan2 or Arc Tangent with 2 parameters

The property sheet of the object is shown below



16.5 *Ceiling*

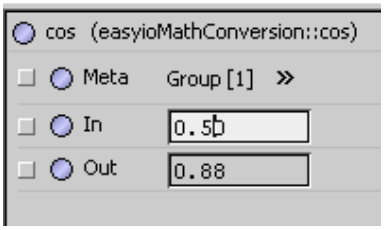
Ceil or Round Up Value object
Returns the smallest integral value that is not less than *x*.
The property sheet of the object is shown below



16.6 *Cosine*

Cos or Cosine object

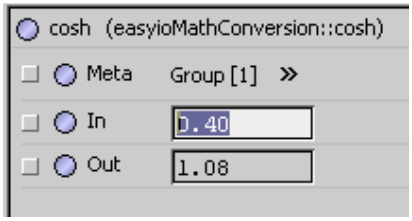
The property sheet of the object is shown below



16.7 *Cosine Hyperbolic*

Cosh or hyperbolic cosine Object

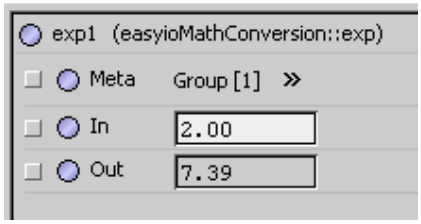
The property sheet of the object is shown below



16.8 *Exponential*

Exp or Exponential Object

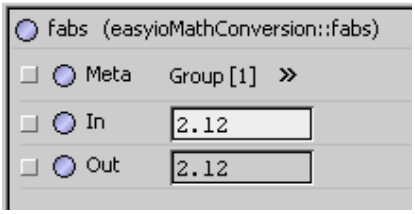
The property sheet of the object is shown below



16.9 *Float Absolute*

Fabs or Absolute value of input

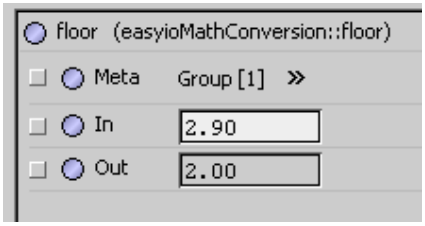
The property sheet of the object is shown below



16.10 *Floor*

Floor or Round Down Value object

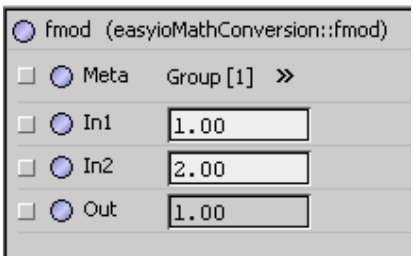
The property sheet of the object is shown below



16.11 *FMod*

Fmod or Remainder of the Division of input

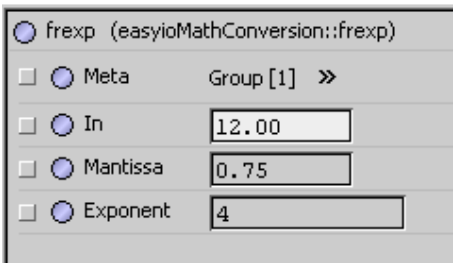
The property sheet of the object is shown below



16.12 *Frexp*

Frexp or Get significand and exponent

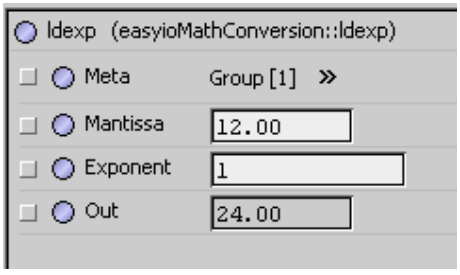
The property sheet of the object is shown below



16.13 *Ldexp*

Ldexp or Generate number from significand and exponent object

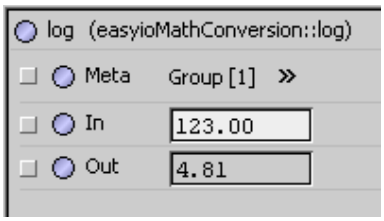
The property sheet of the object is shown below



16.14 *Log*

Log or compute natural logarithm

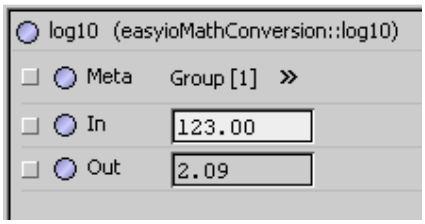
The property sheet of the object is shown below



16.15 *Log 10*

Log10 or compute common logarithm of input object

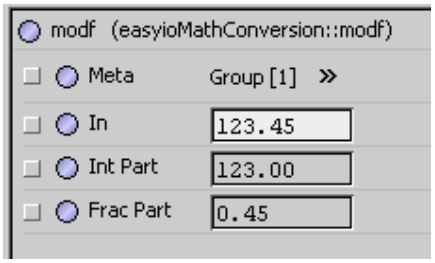
The property sheet of the object is shown below



16.16 *ModF*

Modf or break in fractional and integral parts of the input.

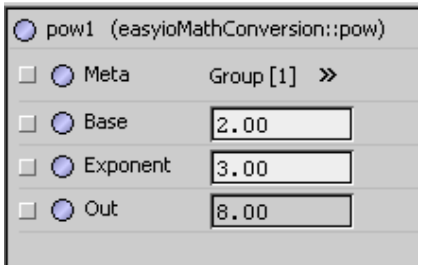
The property sheet of the object is shown below



16.17 *Power*

Pow or raise to power object

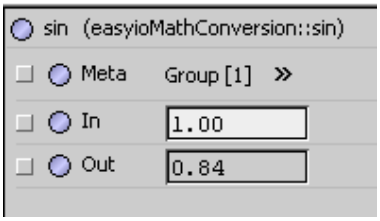
The property sheet of the object is shown below



16.18 *Sine*

Sin or compute sin object

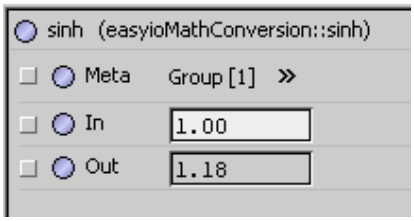
The property sheet of the object is shown below



16.19 *Sine Hyperbolic*

Sinh or hyperbolic sine object

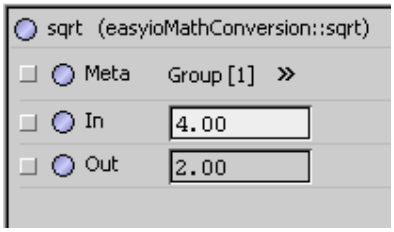
The property sheet of the object is shown below



16.20 *Square Root*

Sqrt or compute square root object

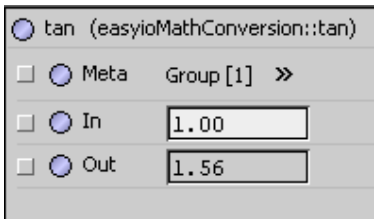
The property sheet of the object is shown below



16.21 *Tangent*

Tan or compute Tangent object

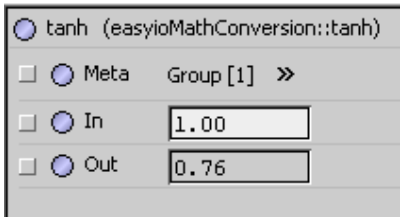
The property sheet of the object is shown below



16.22 *Tangent Hyperbolic*

Tanh or compute hyperbolic tangent object

The property sheet of the object is shown below



17 EasyioModbus

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
16	EasyioModbus	1.0.43.20	Easyio 1.0.43.00 or higher	CoilOutput DiscreteInput HoldingRegFloat HoldingRegLong HoldingRegWord InputRegFloat InputRegLong InputRegWord

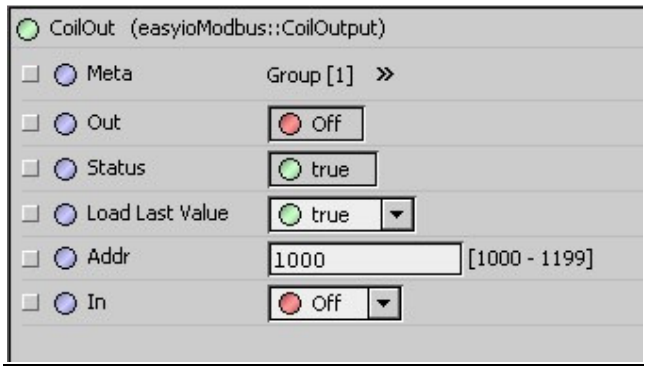
This kit contains 8 objects. All the objects are to be used for Modbus points broadcast. Each object has a max of 200 register.

To use these objects just drag and drop into the wire sheet.

17.1 *CoilOutput*

CoilOutput maximum 200 Coil Output registers can be defined (ID: 1000 - 1199).

The property sheet of the object is shown below



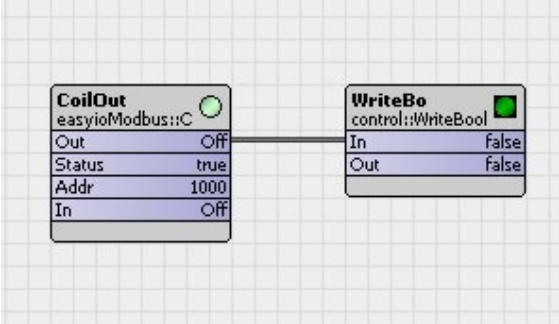
- **Out**
  Coil Output output value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Load Last Value**
  Enable the load last value when program start
  true = load last stored value when start
  false = do not load last value

- **Address**
  Register address
  1000 – 1199

- **In**
  Coil Output input value.

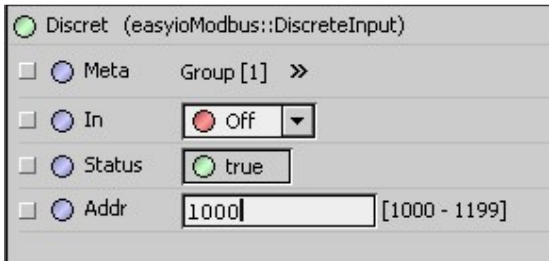


Example of a modbus coil point driving a BooleanWritable object.

17.2 *DiscreteInput*

DiscreteInput maximum 200 DiscreteInput registers can be defined (ID: 1000 - 1199).

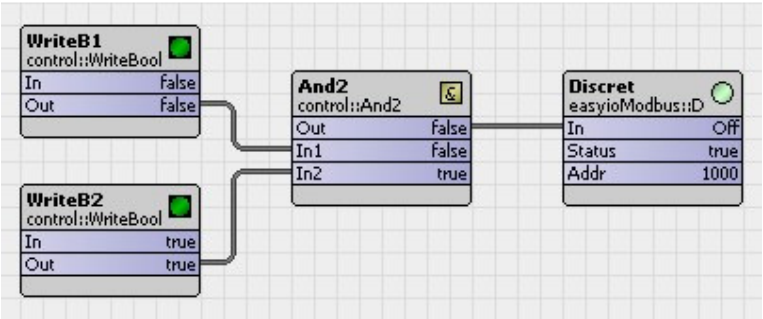
The property sheet of the object is shown below



- **In**
  Coil Output input value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Address**
  Register address
  1000 – 1199



Example of a modbus discrete point reading from an AND object.

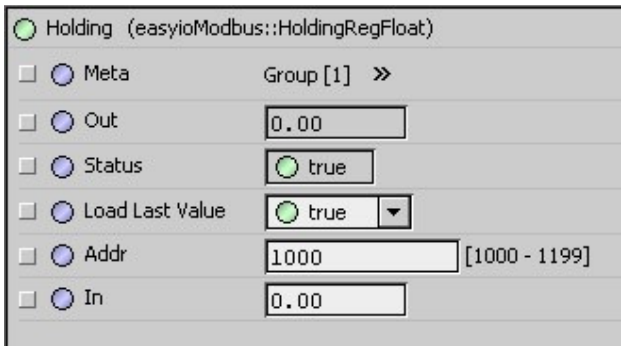
17.3 *Holding Reg Float*

HoldingRegFloat maximum 200 Holding Register can be defined (ID: 1000 - 1199). 200 Holding Register is including floating point, 16-bit data & 32-bit data type. Floating Point & 32-bit data occupy 2 register address where 16-bit occupies 1 register address. All addresses are based-0.

Example:

```
Address     Name                  Register Type
-------     --------------------  --------------
1000        SetPoint              floating point (2 registers)
1002        Duration              32-bit data (2 registers)
1004        Count                 16-bit data (1 register)
1005        Delay                 16-bit data (1 register)
```

The property sheet of the object is shown below



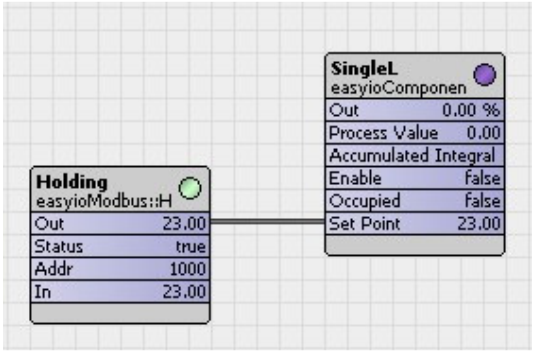
- **Out**
  HoldingRegFloat output value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Load Last Value**
  Enable the load last value when program start
  true = load last stored value when start
  false = do not load last value

- **Address**
  Register address
  1000 – 1199

- **In**
  HoldingRegFloat input value.



Example of a modbus HoldingregFloat writing setpoint to a SingleLoop object.

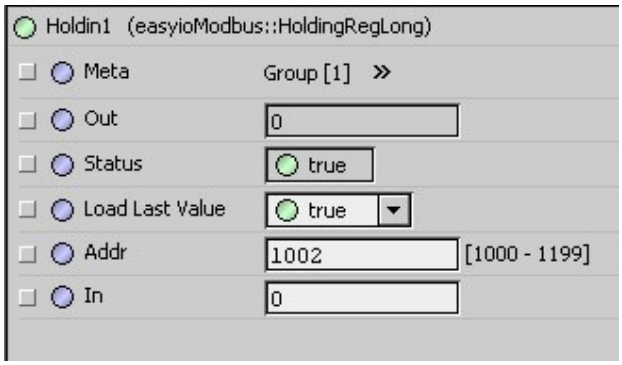
17.4 *Holding Reg Long*

HoldingRegLong maximum 200 Holding Register can be defined (ID: 1000 - 1199). 200 Holding Register is including floating point, 16-bit data & 32-bit data type. Floating Point & 32-bit data occupy 2 register address where 16-bit occupies 1 register address. All addresses are based-0.

Example:

Address	Name	Register Type
1000	SetPoint	floating point (2 registers)
1002	Duration	32-bit data (2 registers)
1004	Count	16-bit data (1 register)
1005	Delay	16-bit data (1 register)

The property sheet of the object is shown below



- **Out**
  HoldingRegLong output value.

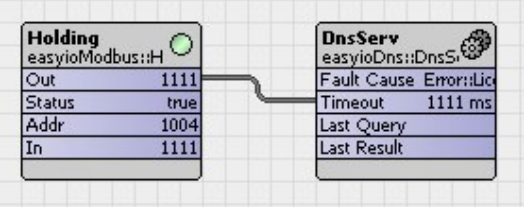
- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Load Last Value**
  Enable the load last value when program start

true = load last stored value when start
false = do not load last value

- **Address**
  Register address
  1000 – 1199

- **In**
  HoldingRegLong input value.



Example of a modbus HoldingRegLong writing year value to a DNS service object.

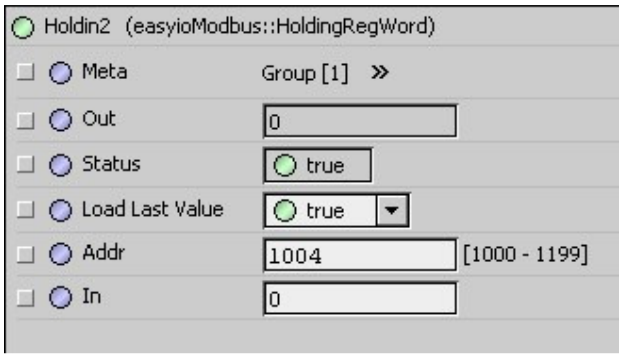
17.5 *Holding Reg Word*

HoldingRegWord maximum 200 Holding Register can be defined (ID: 1000 - 1199). 200 Holding Register is including floating point, 16-bit data & 32-bit data type. Floating Point & 32-bit data occupy 2 register address where 16-bit occupies 1 register address. All addresses are based-0.

Example:

```
Address      Name                  Register Type
-------      --------------------  --------------
1000         SetPoint              floating point (2 registers)
1002         Duration              32-bit data (2 registers)
1004         Count                 16-bit data (1 register)
1005         Delay                 16-bit data (1 register)
```

The property sheet of the object is shown below



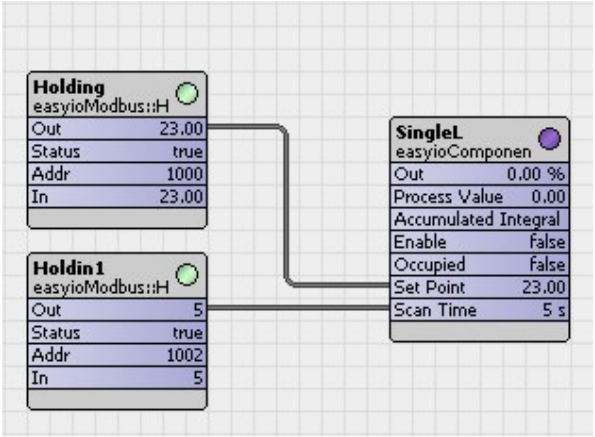
- **Out**
  HoldingRegWord output value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Load Last Value**
  Enable the load last value when program start
  true = load last stored value when start
  false = do not load last value

- **Address**
  Register address
  1000 – 1199

- **In**
  HoldingRegWord input value.



Example of a modbus HoldingRegWord writing Scan Time to a SingleLoop object.

17.6 *Input Reg Float*

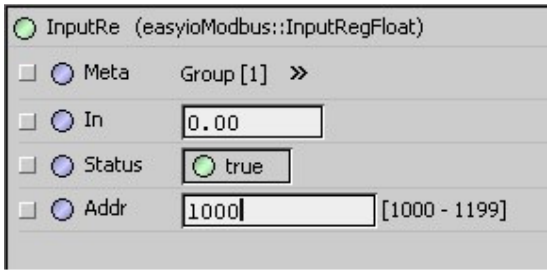
InputRegFloat maximum 200 Input Register can be defined (ID: 1000 - 1199). 200 Holding Register is including floating point, 16-bit data & 32-bit data type. Floating Point & 32-bit data occupy 2 register address where 16-bit occupies 1 register address. All addresses are based-0.

Example:

```
Address      Name                    Register Type
-------      --------------------    --------------
1000         SetPoint                floating point (2 registers)
```

```
1002        Duration                32-bit data (2 registers)
1004        Count                   16-bit data (1 register)
1005        Delay                   16-bit data (1 register)
```

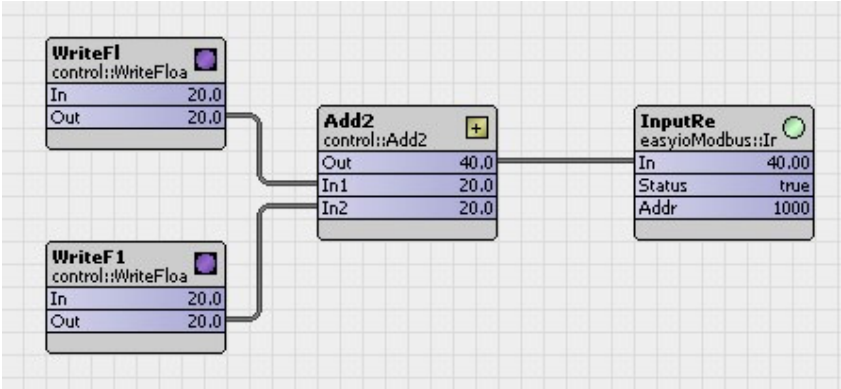
The property sheet of the object is shown below



- **In**
  Coil Output input value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Address**
  Register address
  1000 – 1199



Example of a modbus InputRegFloat reading Out from a Add object.

17.7 *Input Reg Long*

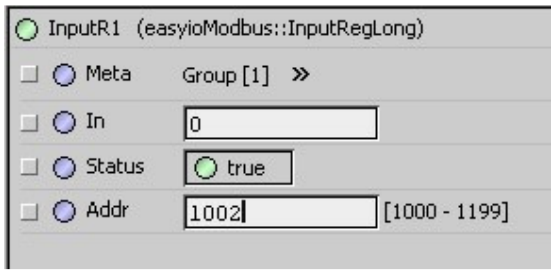
InputRegLong maximum 200 Input Register can be defined (ID: 1000 - 1199). 200 Holding Register is including floating point, 16-bit data & 32-bit data type. Floating Point & 32-bit data occupy 2 register address where 16-bit occupies 1 register address. All addresses are based-0.

Example:

```
Address     Name                    Register Type
-------     --------------------    --------------
```

```
1000        SetPoint                floating point (2 registers)
1002        Duration                32-bit data (2 registers)
1004        Count                   16-bit data (1 register)
1005        Delay                   16-bit data (1 register)
```

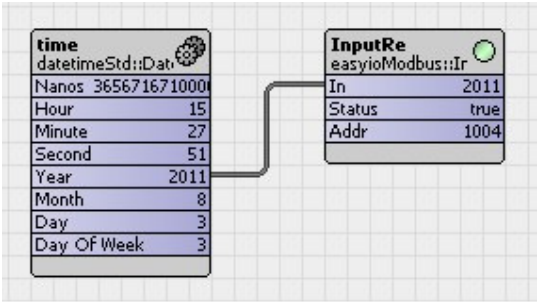
The property sheet of the object is shown below



- **In**
  Coil Output input value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Address**
  Register address
  1000 – 1199



Example of a modbus InputRegLong reading current year from a Date object.

17.8 *Input Reg Word*

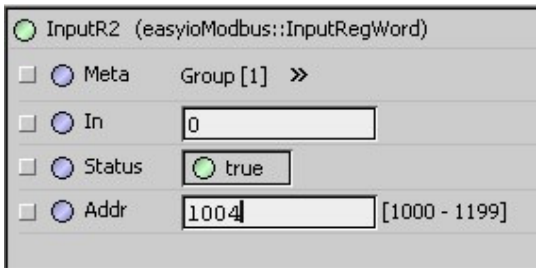
InputRegWord maximum 200 Input Register can be defined (ID: 1000 - 1199). 200 Holding Register is including floating point, 16-bit data & 32-bit data type. Floating Point & 32-bit data occupy 2 register address where 16-bit occupies 1 register address. All addresses are based-0.

Example:

```
Address     Name                    Register Type
-------     --------------------    --------------
1000        SetPoint                floating point (2 registers)
```

```
1002        Duration                32-bit data (2 registers)
1004        Count                   16-bit data (1 register)
1005        Delay                   16-bit data (1 register)
```

The property sheet of the object is shown below



- **In**
  Coil Output input value.

- **Status**
  Register status. Readonly
  true = valid
  Fale = invalid

- **Address**
  Register address
  1000 – 1199

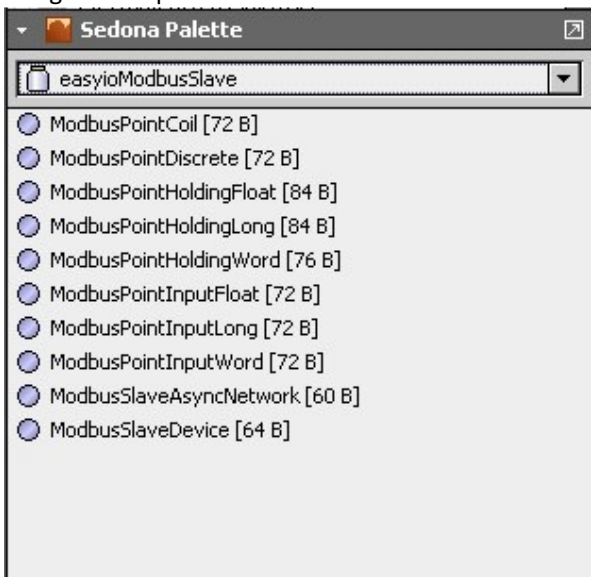
Example of a modbus InputRegWord reading current benchmark cycle per second object.

18 EasyioModbusSlave

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
17	EasyioModbusSlave	1.0.43.21	Easyio 1.0.43.10 or higher	ModbusPointCoil ModbusPointDiscrete ModbusPointHoldingFloat ModbusPointHoldingLong ModbusPointHoldingWord ModbusPointInputFloat ModbusPointInputLong ModbusPointInputWord ModbusSlaveAsyncNetwork ModbusSlaveDevice

This kit contains 10 objects. All the objects are to be used for Modbus Master Slave configuration only.
Modbus Master and Slave configuration is a unique configuration where , the EasyIO 30P Sedona controller as a Modbus Master Controller and Modbus Slave controller connecting to the Rs-485 com port.

To use these objects just drag and drop into the wire sheet.



18.1 *ModbusSlaveAsyncNetwork*

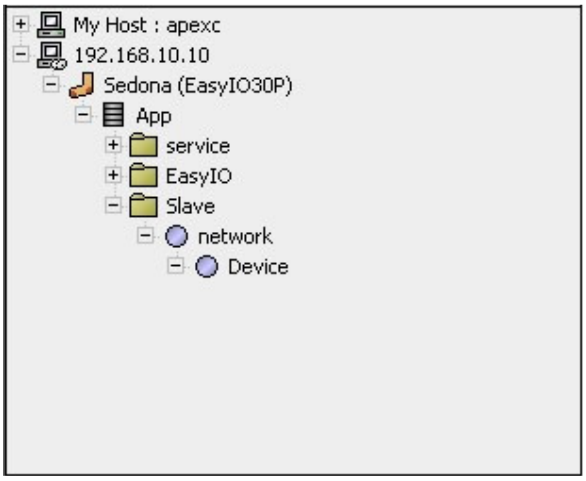
ModbusSlaveAsyncNetwork uses the EasyIO-30P built-in Modbus RTU master driver to connect to Modbus Slave devices.

The serial port has to be disabled first before changing parameter. Do not attempt to change the parameters via web browser.

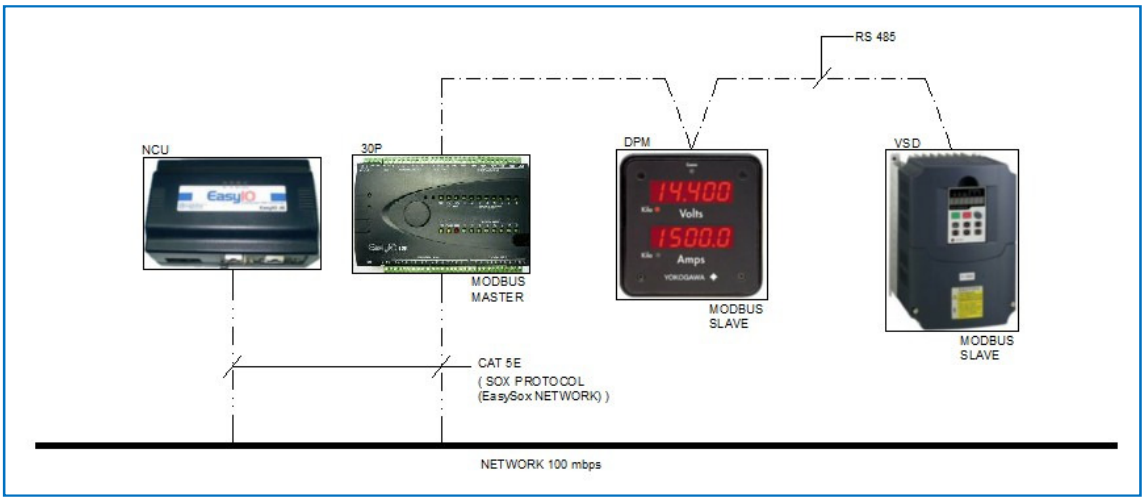
The serial port setting will automatically changed from Modbus to Modbus Master whenever you change the parameter in this object.

****Note: ModbusSlaveAsyncNetwork can be in a new folder or in default EasyIO folder**

Example below show the ModbusSlaveAsyncNetwork is in a new folder created with the name Slave

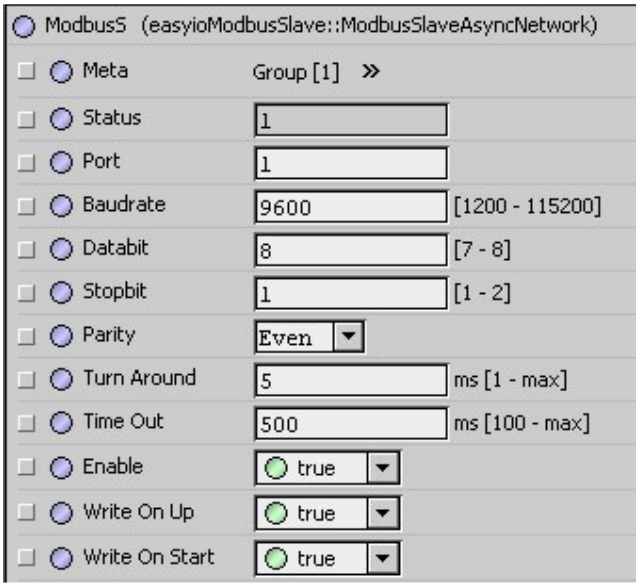


ModbusSlaveAsyncNetwork is under a new folder named Slave.



Typical Modbus Master Slave Configuration.

The property sheet of the object is shown below



- **Status**
  Network status. Readonly
  >0 = OK
  0 = Not connected.
  -1 = port not available
  -2 = invalid baudrate
  -3 = invalid databit
  -4 = invalid stopbit
  -5 = invalid parity

  Normally the value should be “1”

- **Port**
  Serial port, COM1 = 1, COM2 = 2 and etc
  By default the Com port is = “1”

- **Baudrate**
  Serial port baudrate, 1200 – 115200
  Default baud rate set is at 19.2K

  EasyIO30P default baud rate is at 19.2K

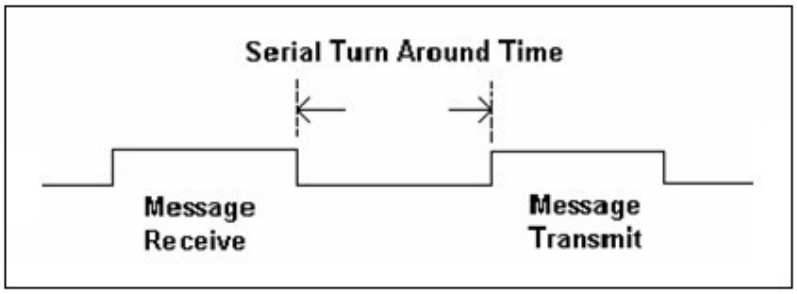
- **Databit**
  Serial port data bit, 7 or 8

- **Stopbit**
  Serial port stop bit, 1 or 2

- **Parity**
  Serial port parity.

```
0 = None
1 = Odd
2 = Even
```

- **Turn Around**
  Turnaround is the time delay in milliseconds between a message can be sent out by driver after it receive the last message.



- **Time Out**
  Modbus Slave device time out in milliseconds.

- **Enable**
  Enable Serial port

- **Write On Up**
  Perform a write whenever the COM/Device is up.

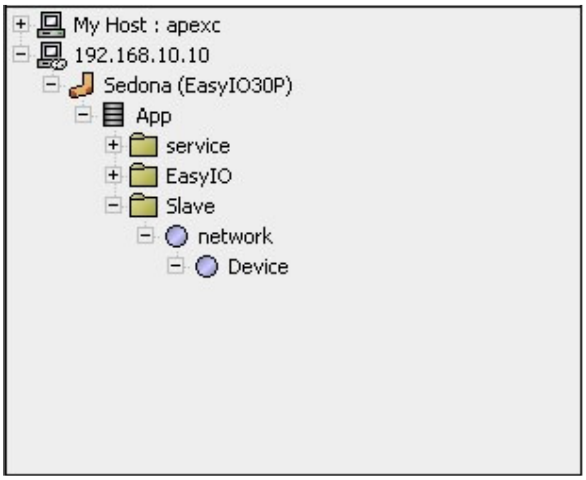
- **Write On Start**
  Perform a write whenever the COM/Device is up.

18.2 *ModbusSlaveDevice*

ModbusSlaveDevice must be child of Modbus Slave Network.It cannot be anywhere else.

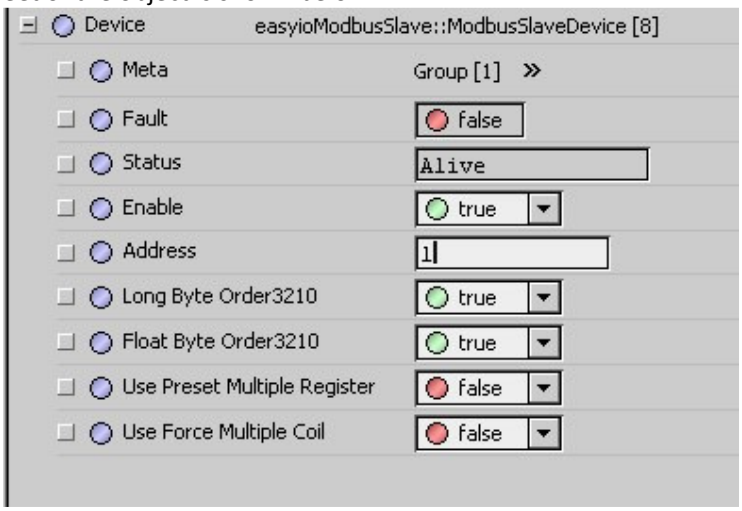
****Note: ModbusSlaveDevice can only be a child of ModbusSlaveNetwork**

Example below show the ModbusSlaveDevice is in a child oof the network.



ModbusSlaveDevicek is a child of ModbusSlaveNetwork.

The property sheet of the object is shown below



- **Fault**
  Fault status. Readonly
  False = No fault
  True = Fault

- **Status**
  Current status. Readonly

```
Alive = Slave Device Alive
Down = Slave Device Down
```

- **Enable**
  Enable device.

- **Address**
  Device Address in decimal

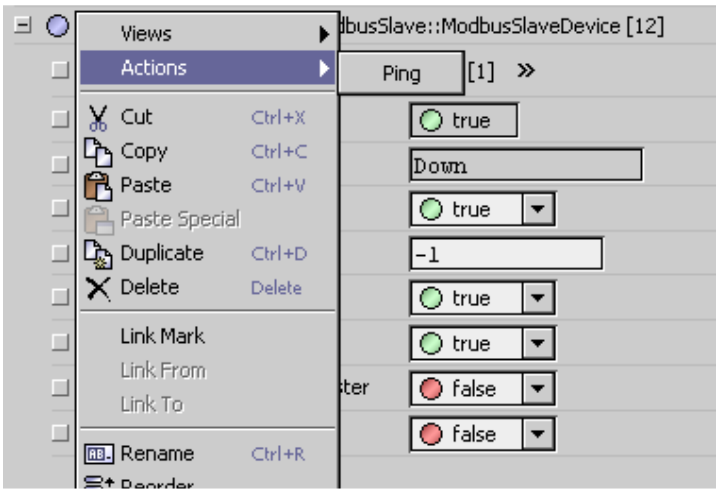
- **Long Byte Order3210**
  Long type data (32-bit) byte order.
  True = 3210F
  False = 1032

- **Float Byte Order3210**
  Floating point type data byte order
  True = 3210
  False = 1032

- **Use Preset Multiple Register**
  True = support Modbus function 16

- **Use Force Multiple Coil**
  True = support Modbus function 15

- **Action**
  Version 1.0.43.21 offers device ping. Right mouse button at the device to do a force ping.

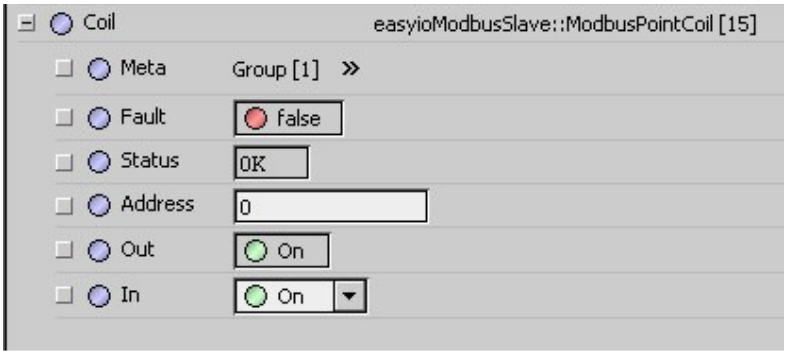


18.3 *ModbusPointCoil*

ModbusPointCoil is Modbus Coil Output Point

****Note: ModbusPointCoil can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below



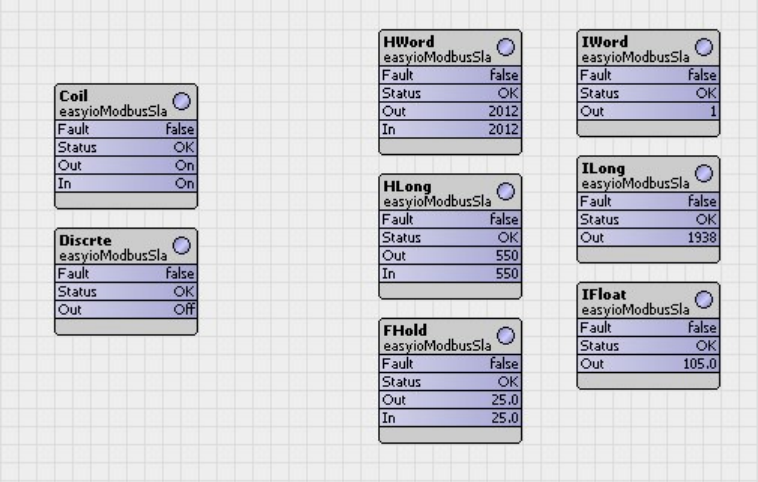
- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly

- **In**
  Local input value.



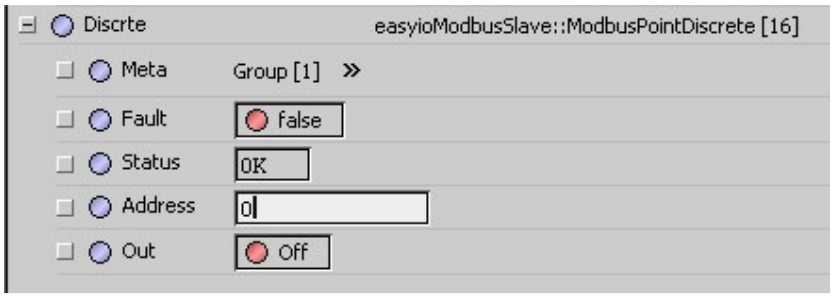
ModbusSlaveDevice register example.

18.4 *ModbusPointDiscrete*

ModbusPointDiscrete is Modbus Discrete Input point

****Note: ModbusPointDiscrete can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below

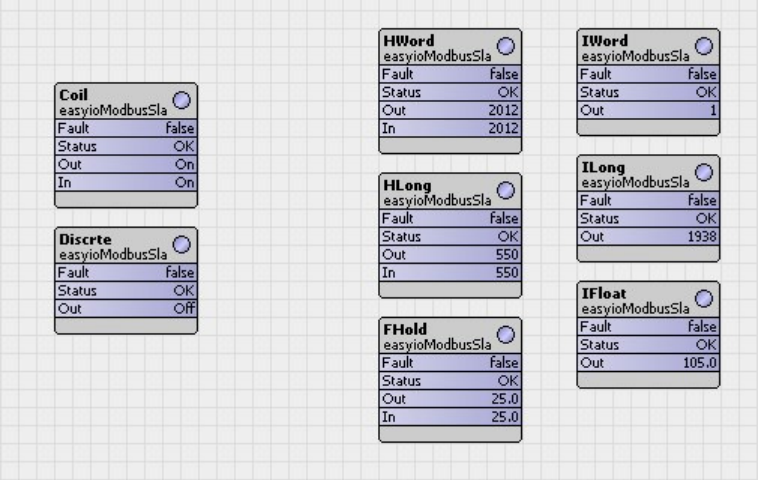


- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly



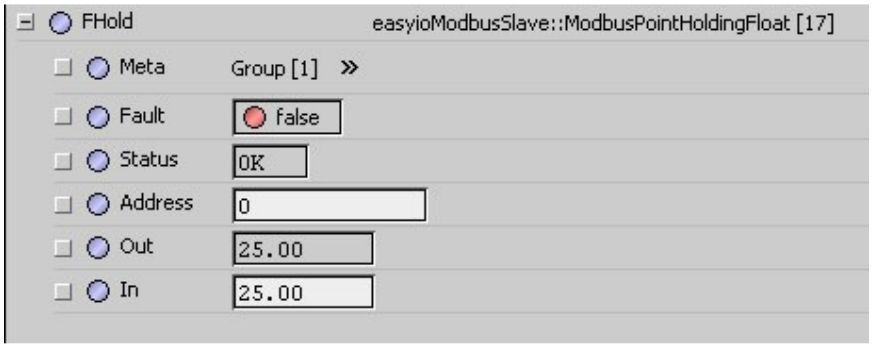
ModbusSlaveDevice register example.

18.5 *ModbusPointHoldingFloat*

ModbusPointHoldingFloat is Modbus Holding Float point

****Note: ModbusPointHoldingFloat can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below



- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

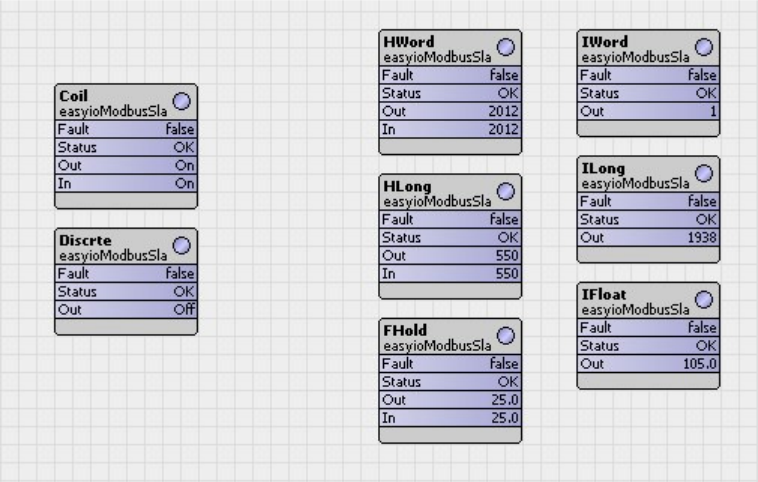
- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**

Modbus register address.
Note that only **Decimal** format is supported.
If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly

- **In**
  Local input value.



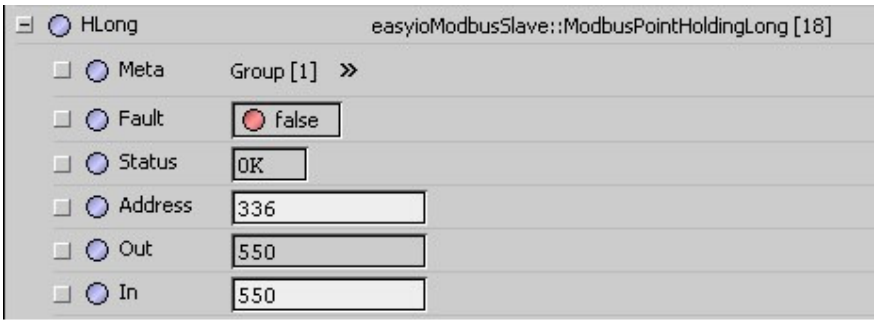
ModbusSlaveDevice register example.

18.6 *ModbusPointHoldingLong*

ModbusPointHoldingLong is Modbus Holding Long point

****Note: ModbusPointHoldingLong can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below



- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

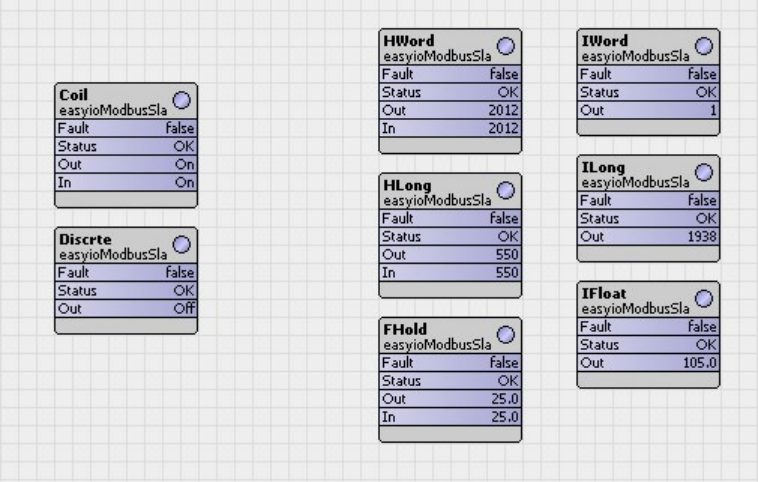
- **Status**
  Status of the point

OK = Online
Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly

- **In**
  Local input value.



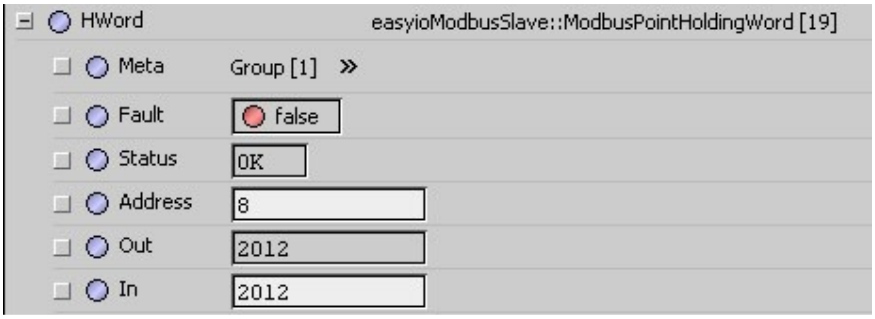
ModbusSlaveDevice register example.

18.7 *ModbusPointHoldingWord*

ModbusPointHoldingWord is Modbus Holding Word point

****Note: ModbusPointHoldingWord can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below



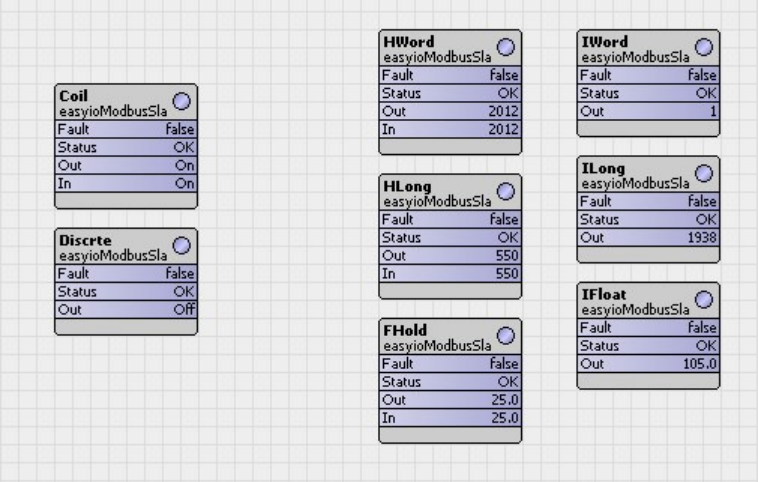
- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly

- **In**
  Local input value.



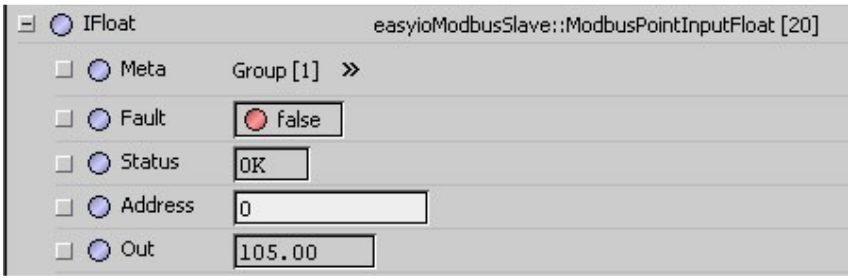
ModbusSlaveDevice register example.

18.8 *ModbusPointInputFloat*

ModbusPointInputFloat is Modbus Input Float point

****Note: ModbusPointInputFloat can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below

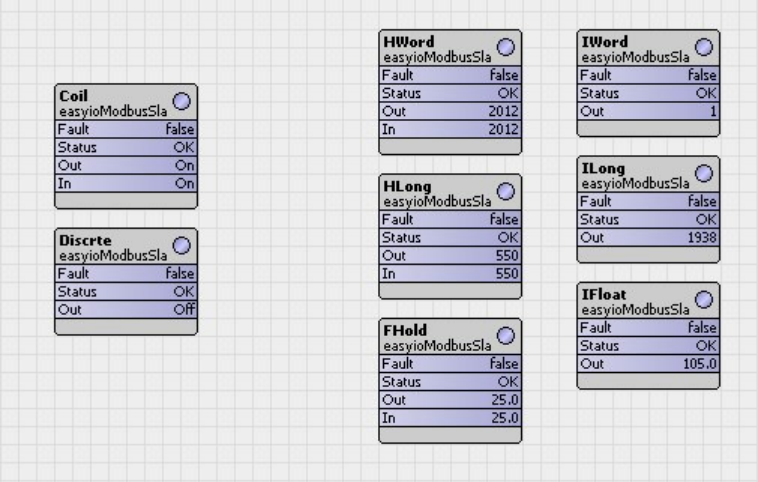


- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly



ModbusSlaveDevice register example.

18.9 *ModbusPointInputLong*

ModbusPointInputLong is Modbus Input Long point

****Note: ModbusPointInputLong can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below

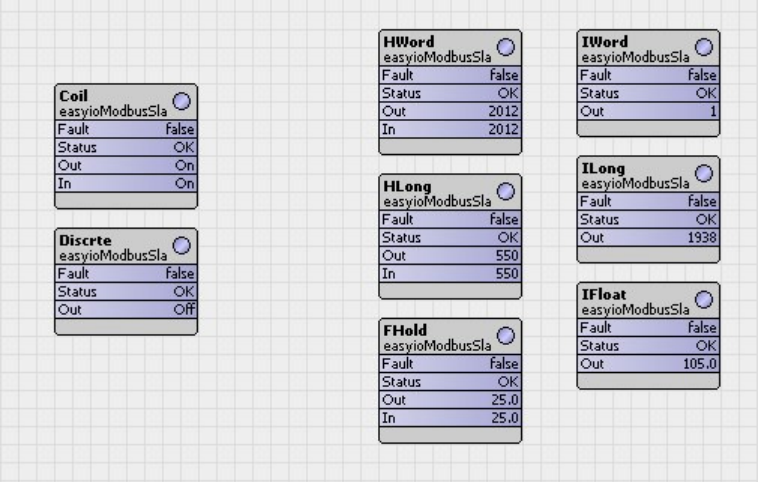
- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**

Current Coil Output state. Readonly



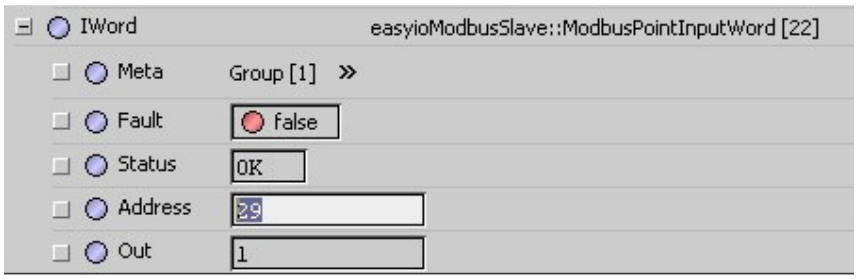
ModbusSlaveDevice register example.

18.10 *ModbusPointInputWord*

ModbusPointInputWord is Modbus Input Word point

****Note: ModbusPointInputWord can only be a child of ModbusSlaveDevice**

The property sheet of the object is shown below

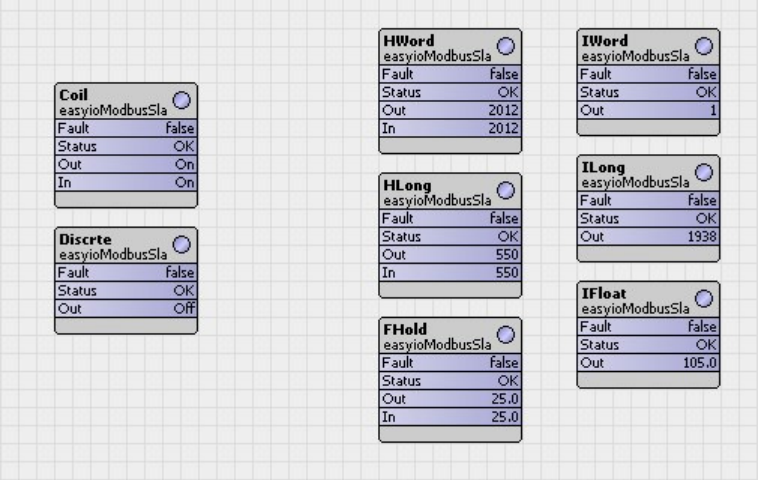


- **Fault**
  Status of the Modbus register.
  false = Valid
  true =Invalid

- **Status**
  Status of the point
  OK = Online
  Down = Offline

- **Address**
  Modbus register address.
  Note that only **Decimal** format is supported.
  If the Modbus device register is in HEX , need to convert to DEC.

- **Out**
  Current Coil Output state. Readonly



ModbusSlaveDevice register example.

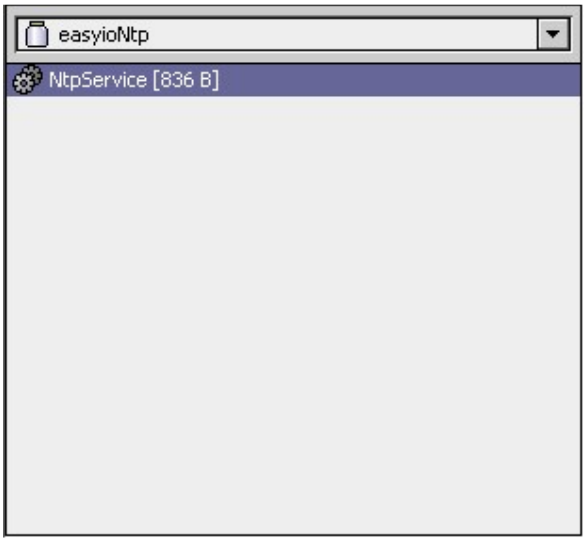
19 EasyioNTP

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
17	easyioNtp	1.0.45	Easyio 1.0.43.10 or higher easyioDns 1.0.45 or higher easyioLicense 1.0.45 or higher	NtpService

easyioNtp kit provide the Time Synchronization Service to the devices, over an Internet enabled network, from a user defined time server.

EasyioNtp kit contains 1 object :

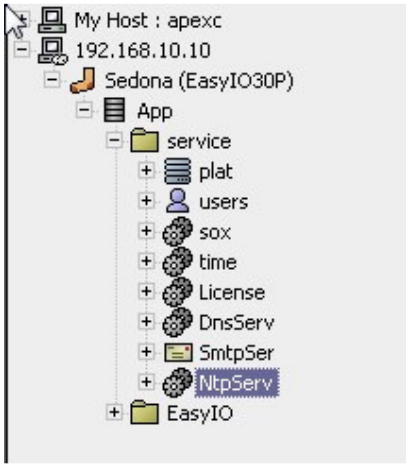
To use these objects just drag and drop into the wire sheet.



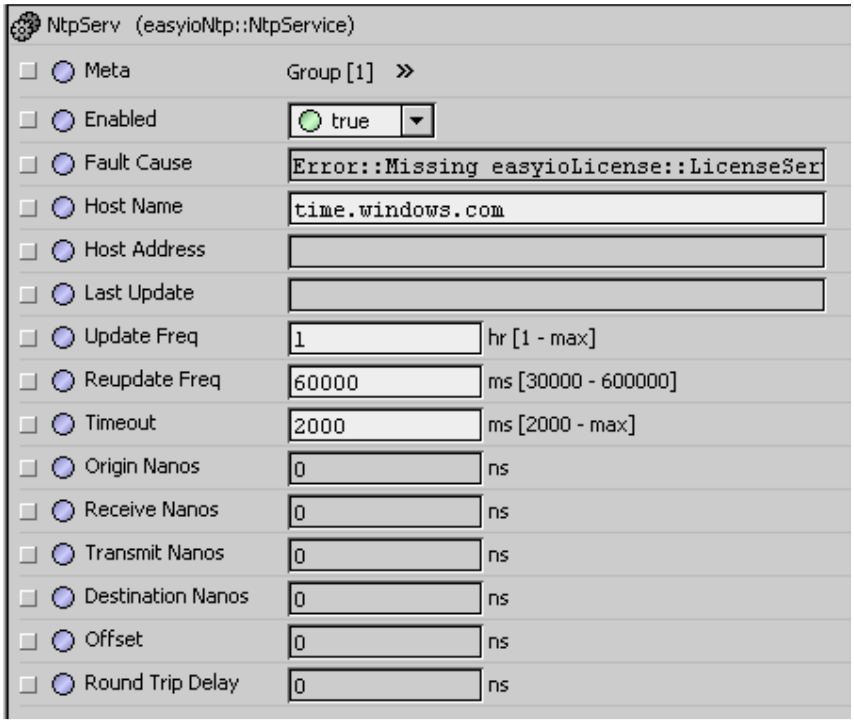
19.1 *NtpService*

NtpService is a service that provides the device with the time synchronization service. This service is build by reference on Network Time Protocol (NTP) version 3 documentation (rfc1305).

****Note: NtpService must be drop inside Service folder (Sedona -> App -> Service).**



The property sheet of the object is shown below



- **Enabled**
  To enable or disable the NtpService, which is set by user.

- **Fault Cause**
  To show cause of the error, when there was NtpService failure.

- **Host Name**
  The Time Server Host, which will provide the time information for time synchronization. Example: time.windows.com

- **Host Address**
  The IP address corresponding to the Host Name; as a result return by DnsService.

- **Last Update**
  This parameter will show the last successful update time.

- **Update Freq**
  The NtpService update frequency, in the unit of Hour (hr). Default to its minimum value, 1 hr. Value of 1 hr means request will be sent every 1 hr.

- **Reupdate Freq**
  The NtpService re-update frequency when the first attempt of update is failed, in the unit of milliseconds (ms). Default to 60000ms, range from 30000ms to 600000ms.

- **Timeout**
  User defined time length to wait, before the NtpService getting a response, in milliseconds (ms). Default to its minimum value, 2000ms.

- **Origin Nanos**
  The time in the unit of nanoseconds (ns), when the NtpService request sending to time server (Host Address).

- **Receive Nanos**
  The time in the unit of nanoseconds (ns), when the time server is received the request.

- **Transmit Nanos**
  The time in the unit of nanoseconds (ns), when the time server response is sending back to the device (send the request).

- **Destination Nanos**
  The time in the unit of nanoseconds (ns), when the NtpService is received the response from time server.

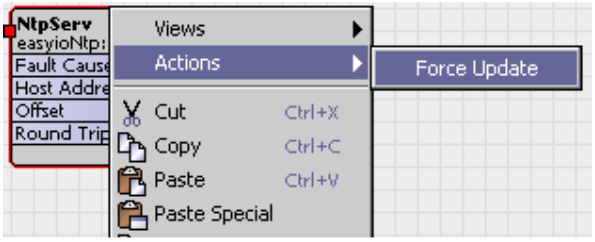
- **Offset**
  Offset is time difference between the device and the time server, which is also the time to be adjusted by device. Value can be either positive or negative.

  Offset = ((Receive Nanos-Origin Nanos) + (Transmit Nanos-Destination Nanos)) / 2

- **Round Trip Delay**
  The total length of time it takes for the request to be sent and the response to be received from time server.

  Round Trip Delay =
  (Destination Nanos-Origin Nanos) - (Transmit Nanos-Receive Nanos)

- **Force Update**



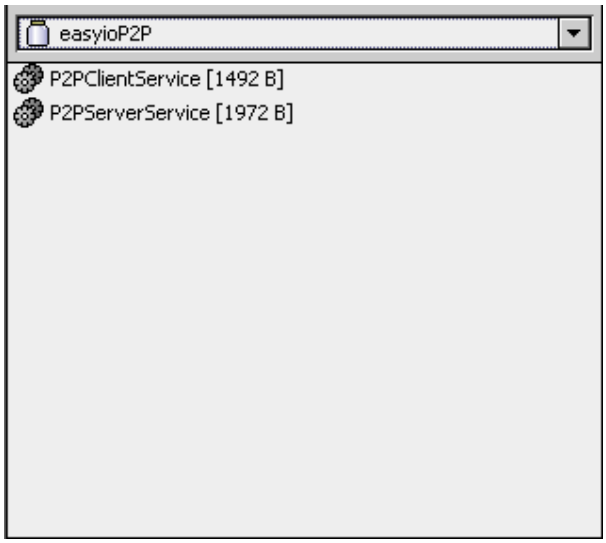
Force the NtpService to send the update request to time server instead of waiting for auto update.

20 EasyioP2P

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
18	easyioP2P	1.0.45	Easyio 1.0.43.0 or higher	P2P Client Service P2P Server Service

This kit contains 2 objects as show below.

By default an EasyIO Sedona controller comes pre-installed with this kit. To use this object just drag and drop into the wire sheet space.

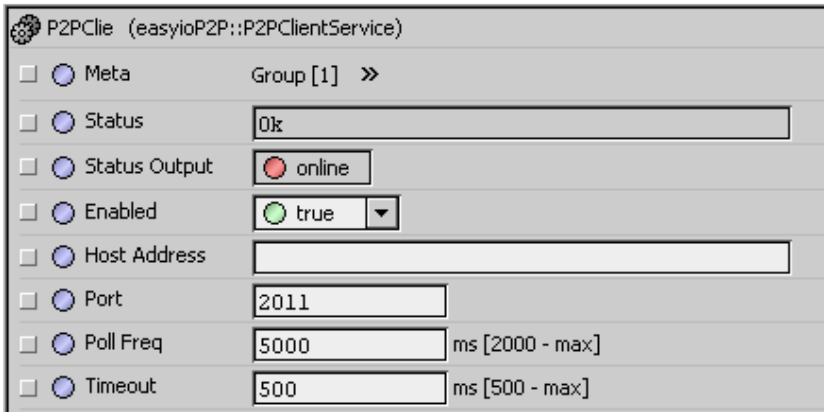


20.1 *P2P Client Service*

P2PClientService is an object where it sits in the client controller polling values from a P2PServer controller.

This object contain 8 boolean output , 8 float output and 8 enum output. These outputs are used to poll value from the user define server controller.

The property sheet of the object is shown below



- **Status**
  This parameter will show the connection between the client and server .

- **Status Output**
  This parameter will show the connection between the client and server.
  It gives user an Boolean output.

  Online = P2P link good
  Down = P2P link no good

- **Enable**
  P2P client polling can be enable or disable with this parameter.

- **Host Address**
  This is the server IP address.

- **Port**
  By default the server port is 2011. However it can be change to other port if it is occupy by other system.

- **Poll Freq**
  By default polling frequency is set to 5 seconds.
  Min poll frequency is 2sec and max is unlimited.

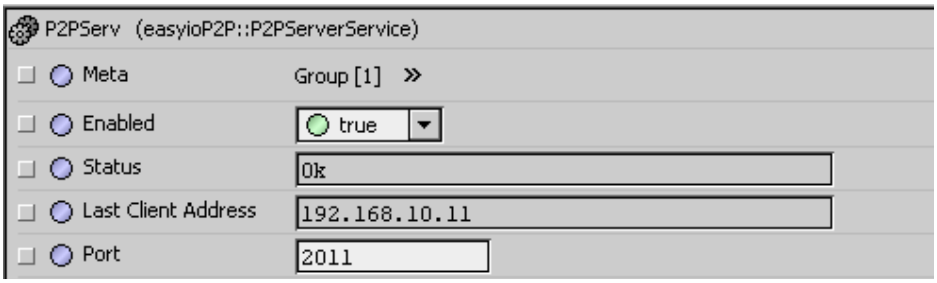
- **Timeout**
  Time period to wait for a response from the server before it time out.

20.2 *P2P Server Service*

P2PServerService is an object where it sits in the client controller polling values from a P2PServer controller.

This object contain 8 boolean output , 8 float output and 8 enum output. These outputs are used to poll value from the user define server controller.

The property sheet of the object is shown below



- **Enable**
  P2P server sending can be enable or disable with this parameter.

- **Status**
  Status of the server object. It will show error when another server object is created with the same port. It will show “cannot binid to port.”

- **Last Client Address**
  Shows the latest client connected to the server object.

- **Port**
  By default the server port is 2011. However it can be change to other port if it is occupy by other system.

21 EasyioPersistentControl

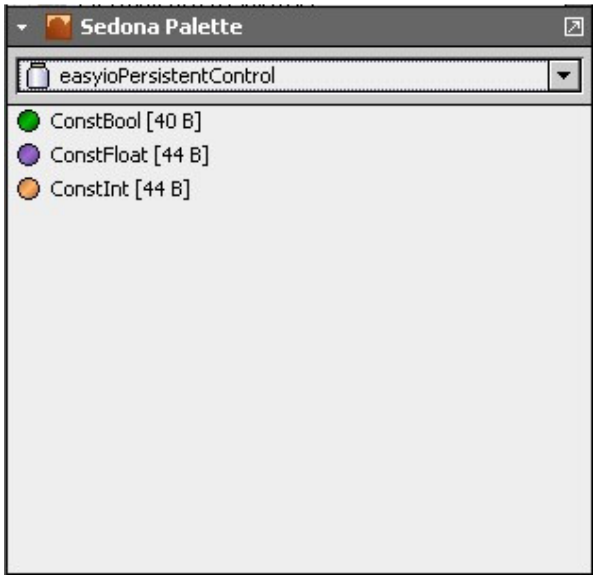
Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
19	EasyioPersistentControl	1.0.45.22	Easyio 1.0.43.10 or higher Platform Easyio 1.0.45.20	ConstBool ConstFloat ConstInt

This kit contains 3 objects. All the objects are to be used for engineer the Sedona apps.

These object are extension from the Tridium Control kit objects (ConstBool,COnstFloat and ConstInt). Tridium constant objects do not save the last written value.

These objects are used if the last value is to remember. For example, a setpoint value changes from a value to another. If a Constant Float from the control.kit is used, it won't remember the last value.

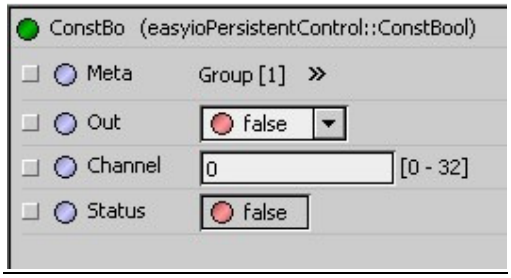
To use these objects just drag and drop into the wire sheet.



21.1 *Constant Boolean*

ConstBool is an object for a Boolean value. This object is used to remember the last state of the object if there is power failure or unsaved apps.

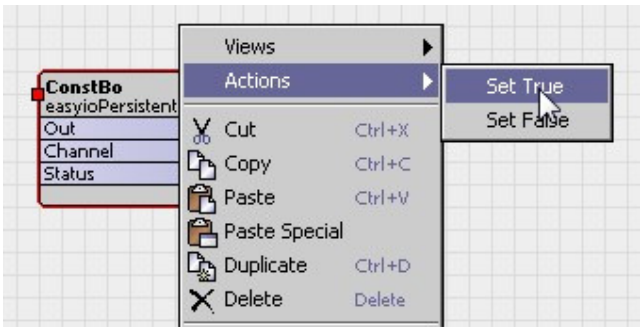
The property sheet of the object is shown below



- **Out**
  The output value for the Constant Boolean

- **Channel**
  Total max of 32 channel can be used. Range from 1-32.

- **Status**
  Status of the object.
  True = valid
  False = Invalid

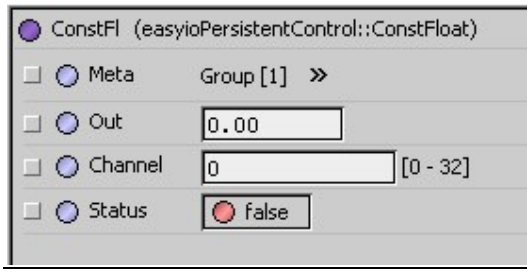


Only a "SET" action will automatically save the value. A link destination will not save the value

21.2 *Constant Float*

ConstFloat is an object for a Float value. This object is used to remember the last state of the object if there is power failure or unsaved apps.

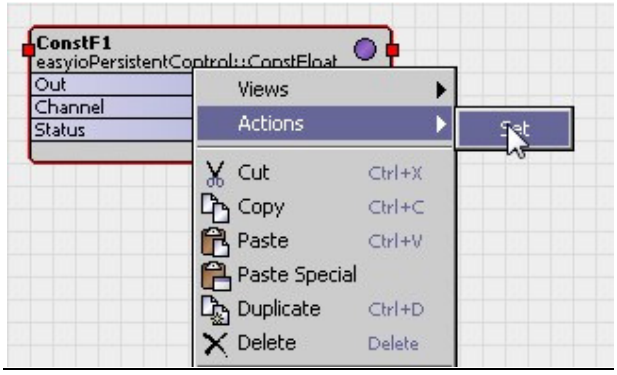
The property sheet of the object is shown below



- **Out**
  The output value for the Constant Float

- **Channel**
  Total max of 32 channel can be used. Range from 1-32.

- **Status**
  Status of the object.
  True = valid
  False = Invalid

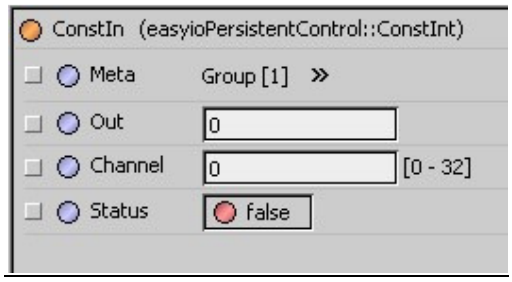


Only a "SET" action will automatically save the value. A link destination will not save the value

21.3 *Constant Integer*

ConstInt is an object for a Integer value. This object is used to remember the last state of the object if there is power failure or unsaved apps.

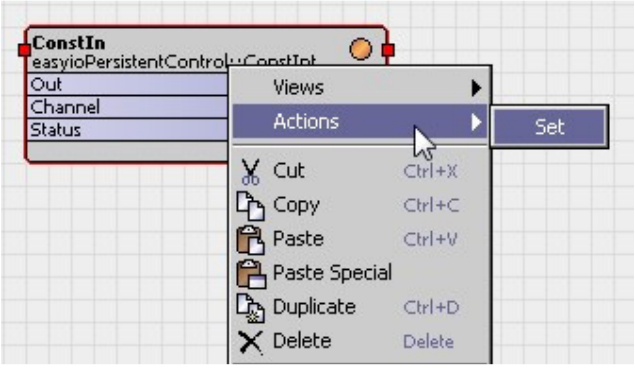
The property sheet of the object is shown below



- **Out**
  The output value for the Constant Boolean

- **Channel**
  Total max of 32 channel can be used. Range from 1-32.

- **Status**
  Status of the object.
  True = valid
  False = Invalid



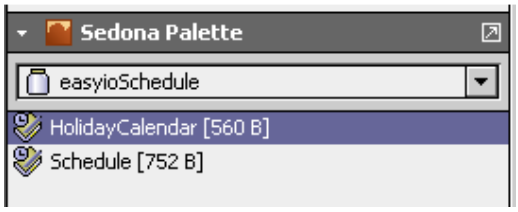
Only a "SET" action will automatically save the value. A link destination will not save the value

22 EasyioSchedule

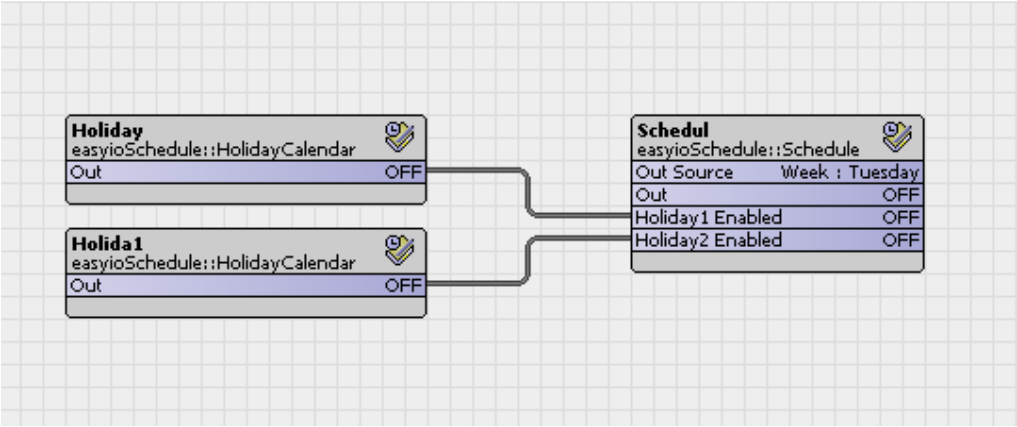
Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
20	EasyioSchedule	1.0.45.2	Easyio 1.0.43.10 or higher	HolidayCalendar Schedule

EasyioScedule kit is built to provide the feature of scheduler. User may customize their schedule for a total 7 weekdays and 2 holidays, while each day having 2 sessions.

EasyioSchdule contains 2 components:



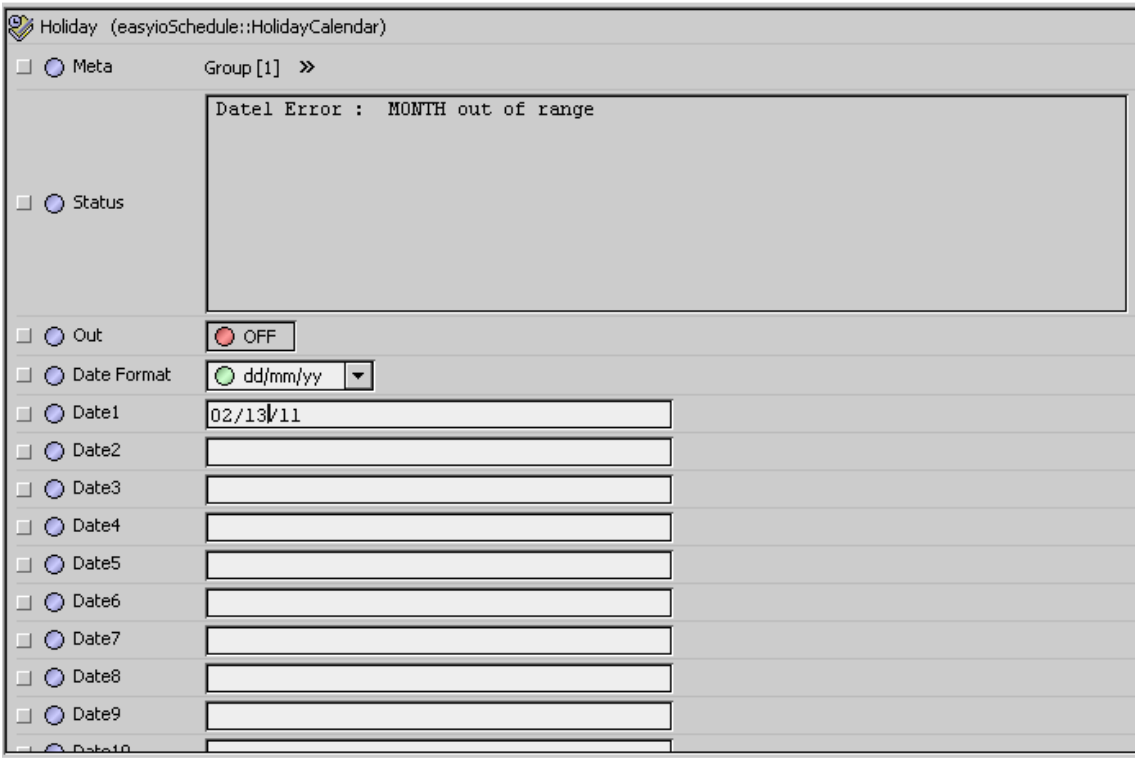
The basic structure of implementing **EasyioSchdule** as below:



22.1 *Holiday Calendar*

HolidayCalendar is used to specify up to 16 holiday dates. It allowed input date in the format either “dd/mm/yy” or “mm/dd/yy”. It’s normally implement by link its output to Schedule property, either Holiday1Enabled or Holiday2Enabled**.**

The property sheet of the object is shown below

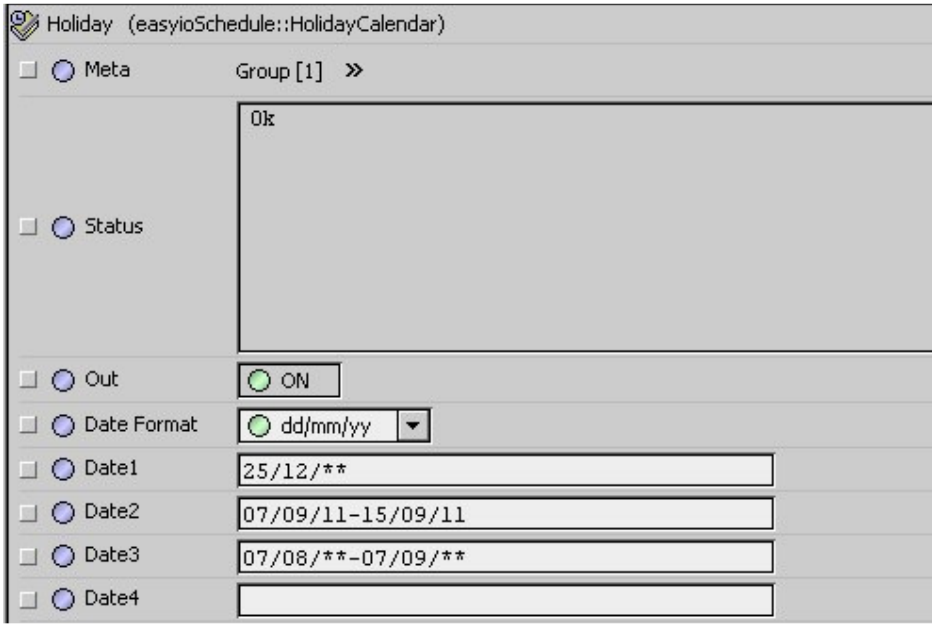


- **Status**
  To show current status of the HolidayCalendar, either indicate with “Ok” or relevant error messages.
  If any error message shown, all Boolean outputs will always set to false, until all error is corrected by user.

- **Out**
  HolidayCalendar’s output. True when current date matched with any of the 16 dates.

- **Date Format**
  Date format that is currently in used, for user to enter the date and for component to read the date entered.
  Two date format allowed: “dd/mm/yy” or “mm/dd/yy”.
  dd = day, mm = month, yy = year

- **Date1**
  Date setting for Holiday 1. Support Date range as well. Refer to example below.
- **Date2**
  Date setting for Holiday 2. Support Date range as well. Refer to example below.
- **Date3**
  Date setting for Holiday 3. Support Date range as well. Refer to example below.
- **Date4**
  Date setting for Holiday 4. Support Date range as well. Refer to example below.
- **Date5**
  Date setting for Holiday 5. Support Date range as well. Refer to example below.
- **Date6**
  Date setting for Holiday 6. Support Date range as well. Refer to example below.
- **Date7**
  Date setting for Holiday 7. Support Date range as well. Refer to example below.
- **Date8**
  Date setting for Holiday 8. Support Date range as well. Refer to example below.
- **Date9**
  Date setting for Holiday 9. Support Date range as well. Refer to example below.
- **Date10**
  Date setting for Holiday 10. Support Date range as well. Refer to example below.
- **Date11**
  Date setting for Holiday 11. Support Date range as well. Refer to example below.
- **Date12**
  Date setting for Holiday 12. Support Date range as well. Refer to example below.
- **Date13**
  Date setting for Holiday 13. Support Date range as well. Refer to example below.
- **Date14**
  Date setting for Holiday 14. Support Date range as well. Refer to example below.
- **Date15**
  Date setting for Holiday 15. Support Date range as well. Refer to example below.
- **Date16**
  Date setting for Holiday 16. Support Date range as well. Refer to example below.



Example of setting the Holiday Schedule.

Date 1 : Example of setting a single date that will occur every year. Noticed that year is set to wild card.

Date 2: Example of setting a date range. Holiday will occur from 7th Sept 2011 till 15th Sept 2011. This is very useful for school holidays. This date range will only occur once.

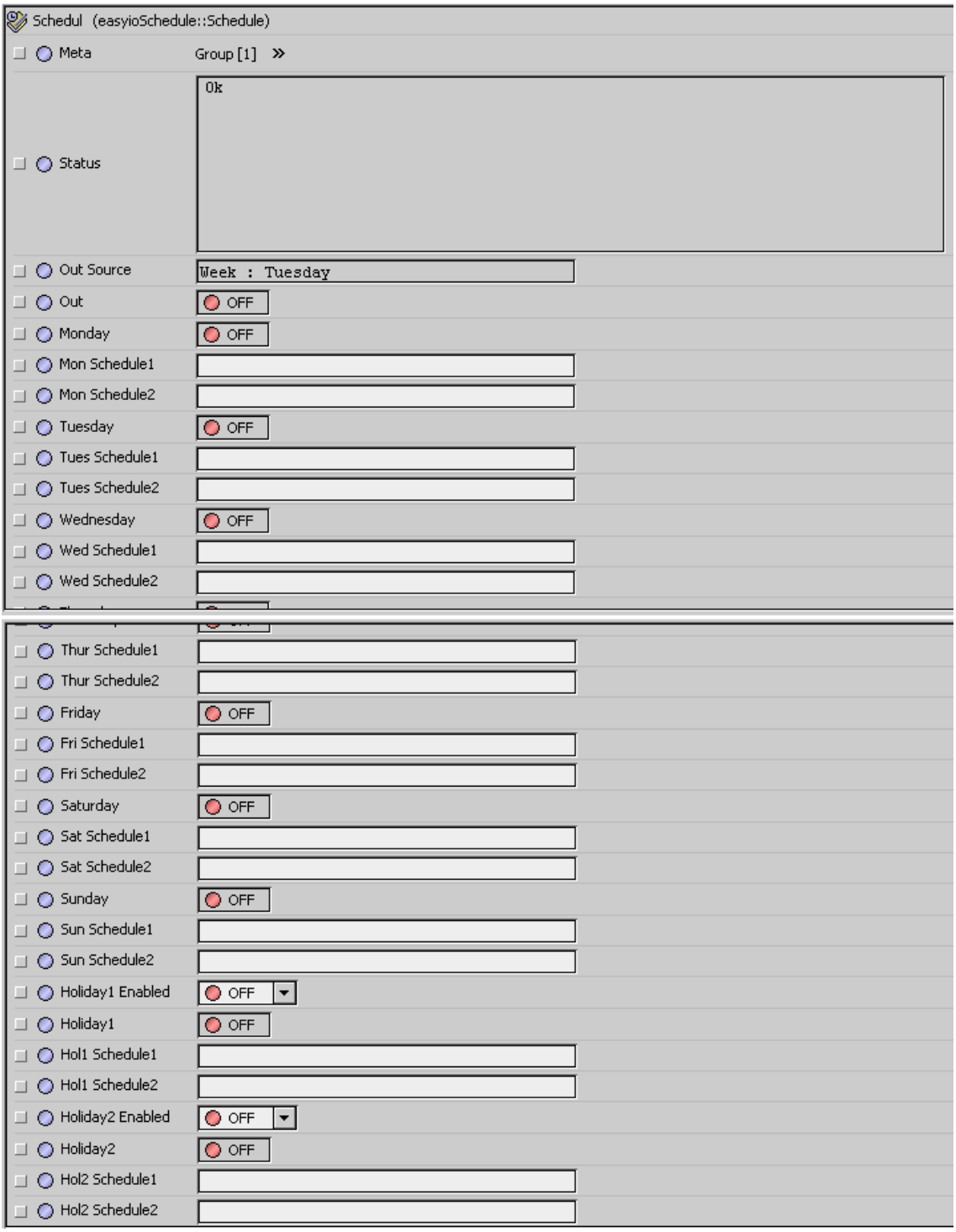
Date 3: Example of setting a date range with wild card , every year from 7th August till 7th Sept will be a holiday.

Holiday Schedule wild card applicable to date range and single date, but is limited as follow.

Single Date wild card : applicable to dd/mm/yy
Date range wild card : applicable to only mm/yy

22.2 *Schedule*

Schedule is used to produce scheduler output, according to the scheduled time setting.
Schedule can setting up to 7 weekdays (Monday to Sunday) and 2 holidays, while each day having 2 sessions.
The property sheet of the object is shown below



- **Status**
  To show current status of the **Schedule**, either indicate with "Ok" or relevant error messages.
  If any error message shown, all Boolean outputs will always set to false, until all error is corrected by user.

- **Out Source**
  To indicate the source property, which provide the schedule output, **Out.**

- **Out**
  Schedule's output, indicate either true or false.

- **Monday**
  Monday scheduled output. If current day of week is Monday, **Out** will read this as its value. Always set to False if day of week doesn't match.

- **Mon Schedule 1**
  Monday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Mon Schedule 2**
  Monday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Tuesday**
  Tuesday scheduled output. If current day of week is Tuesday, **Out** will read this as its value. Always set to False if day of week doesn't match.

- **Tues Schedule 1**
  Tuesday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Tues Schedule 2**
  Tuesday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Wednesday**
  Wednesday scheduled output. If current day of week is Wednesday, **Out** will read this as its value. Always set to False if day of week doesn't match.

- **Wed Schedule 1**
  Wednesday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Wed Schedule 2**
  Wednesday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Thursday**
  Thursday scheduled output. If current day of week is Thursday, **Out** will read this as its value. Always show False if day of week doesn't match.

- **Thur Schedule 1**
  Thursday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Thur Schedule 2**
  Thursday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Friday**
  Friday scheduled output. If current day of week is Friday, **Out** will read this as its value. Always show False if day of week doesn't match.

- **Fri Schedule 1**
  Friday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Thur Schedule 2**
  Friday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Saturday**
  Saturday scheduled output. If current day of week is Saturday, **Out** will read this as its value. Always show False if day of week doesn't match.

- **Sat Schedule 1**
  Saturday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Sat Schedule 2**
  Saturday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Sunday**
  Sunday scheduled output. If current day of week is Sunday, **Out** will read this as its value. Always show False if day of week doesn't match.

- **Sun Schedule 1**
  Sunday's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Sun Schedule 2**
  Sunday's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Holiday1 Enabled**
  When set to true, it's in holiday state, and override output from weekdays (Monday to Sunday) and Holiday2. It has the highest priority.
  Usually linked from **HolidayCalendar**'s **Out** property.

- **Holiday1**
  Holiday1 scheduled output. If **Holiday1 Enabled** is ON, **Out** property will read this as its value. Always show False if **Holiday1 Enabled** is OFF.

- **Hol1 Schedule 1**
  Holiday1's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

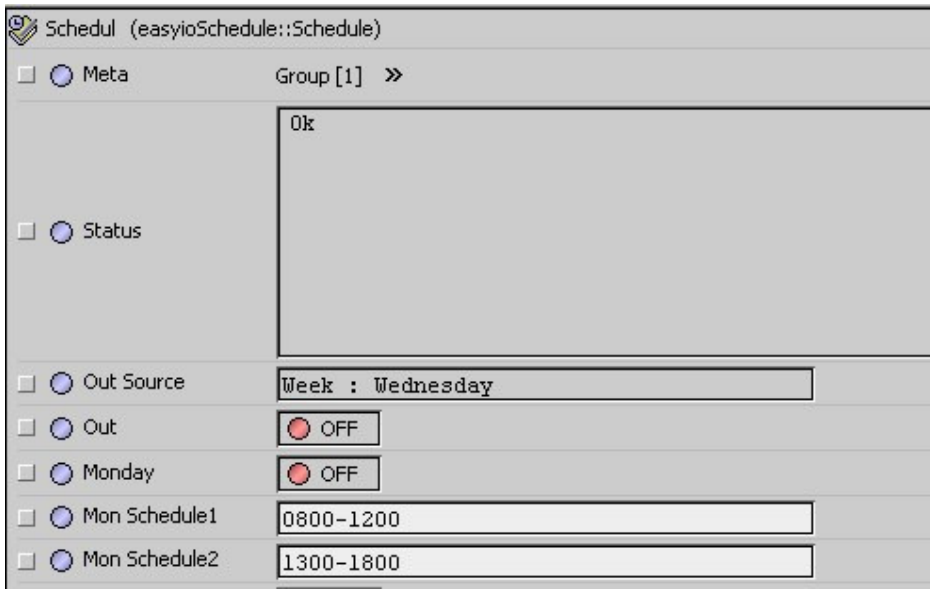
- **Hol1 Schedule 2**
  Holiday1's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Holiday2 Enabled**
  When set to true, it's in holiday state, and override output from weekdays (Monday to Sunday). It has the higher priority than weekdays but lower priority than Holiday1.
  Usually linked from **HolidayCalendar**'s **Out** property.

- **Holiday2**
  Holiday2 scheduled output. If **Holiday2 Enabled** is ON, **Out** property will read this as its value. Always show False if **Holiday2 Enabled** is OFF.

- **Hol2 Schedule 1**
  Holiday2's first session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59

- **Hol2 Schedule 2**
  Holiday2's second session time setting. Format: "HHMM-HHMM".
  HH = 0 – 23, MM = 0 – 59



Example of setting the schedule object.
Time On : 0800-1200
Time Off : 1201-1259
Time On : 1300-1800
Time Off : 1801-2399

23 EasyioSox

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
21	EasyioSox	**Discontinue**	Easyio 1.0.43 or higher	**Discontinue**

Discontinue

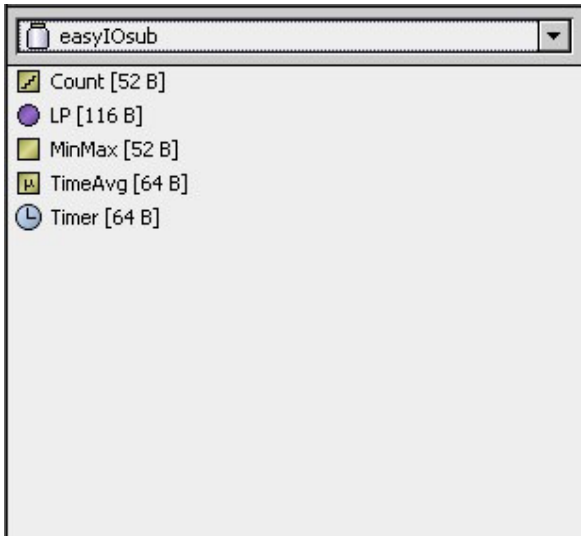
24 EasyioSub

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
22	EasyioSub	1.0.45.00	Easyio 1.0.43 or higher	Count LP MinMax TimeAvg Timer

This kit contains 5 objects. All the objects are to be used for engineer the Sedona apps.

Objects are originally from Tridium. These 5 objects are additional objects derived from Sedona Workbench 1.0.47 control.kit. It is not available in Sedona Workbench 1.0.45.

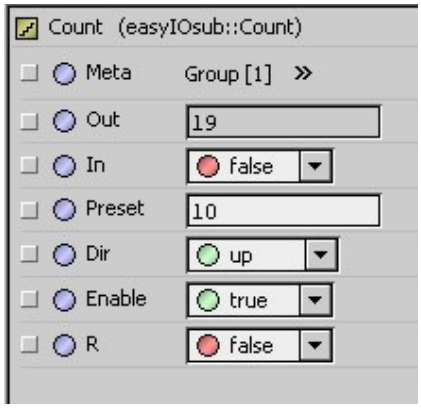
To use these objects just drag and drop into the wire sheet.



24.1 *Count*

Count is an object to count a Boolean value. It can be count **"UP"** or **"DOWN".**

The property sheet of the object is shown below



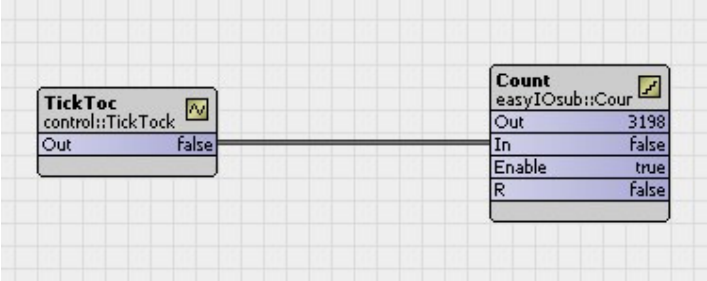
- **Out**
  Number of times "in" property has transitioned from 0 to 1

- **Preset**
  Presets the counter to a specific value, defaults to 0

- **Dir**
  Configures direction. True = "up", False = "down"

- **Enable**
  To enable input

- **R**
  if r is true, out = preset and no counting takes place
  It act as a reset switch

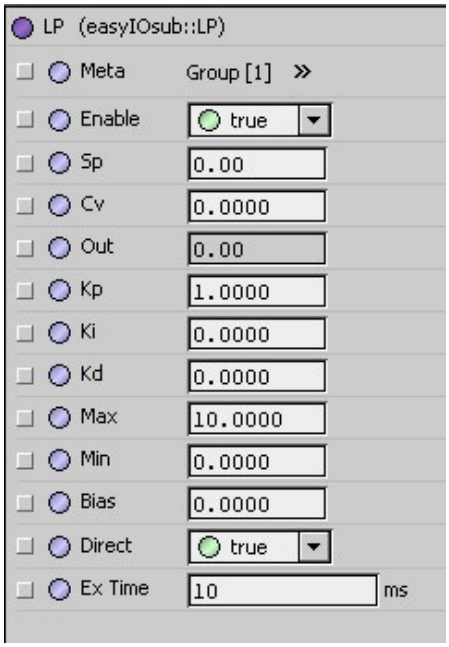


An example of Count object counting transition from a Tick Tock object

24.2 *Loop Point*

LP, PID loop object , this is an updated Loop Point , this fixes the output turn to *"nan"* when a invalid Process Value is detected.

The property sheet of the object is shown below



- **Enable**
  To enable the LP object

- **Sp**
  Setpoint for the LP object

- **Cv**
  Control Variable for the LP object

- **Out**
  The output of the LP object

- **Kp**
  Propotional gain for LP object

- **Ki**
  Integral Gain for LP object

- **Kd**
  Derivative Gain for the LP object

- **Max**

Scale Max for the LP output

- **Min**
  Scale Max for the LP output

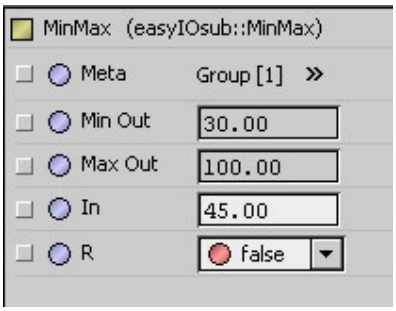
- **Bias**

- **Direct**
  This parameter defines the output action.
  True = cooling
  False = heating

- **Ex Time**
  Defines the interval at which the process variable is sampled or the loop algorithm is executed in seconds.

24.3 *MinMax*

MinMax is an object that will show the Min and Max value of a float input value. It computes min and max of an input value every execute cycle

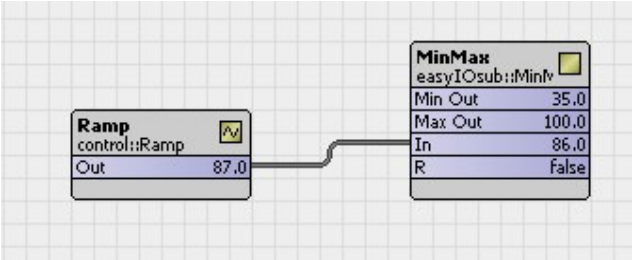


- **Min Out**
  The Min value for the input

- **Max Out**
  The Max value for the input

- **In**
  Input value which is to track

- **R**
  Reset input.
  If R = true, then minOut and maxOut are forced to "in" value.



An example of MinMax object computing the Min Value and Max Value from Ramp object

24.4 *TimeAverage*

TimeAvg object averages "in" over the configured time. The actual time is marked in a resolution of scan period such that number of samples

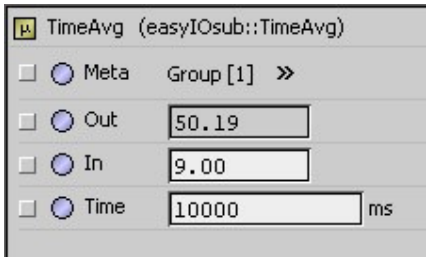
Averaged = time/Sys.app.scanPeriod

Note that this is NOT a running average - this object caches the average over the previous time as the out value, and updates out every "time" ms.

Until a full time cycle has elapsed, the out is set to the average off all samples collected up until that point.

The average may be reset/restarted at any time using the "reset" action.

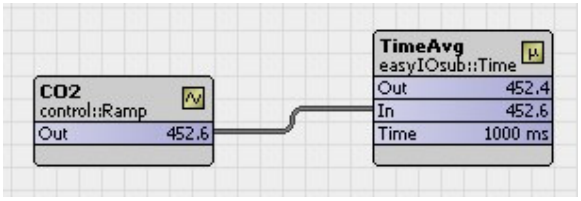
The property sheet of the object is shown below



- **Out**
  Value averaged over last "time" period

- **In**
  The input value to average

- **Time**
  The time period over which to average the in value to get the out value



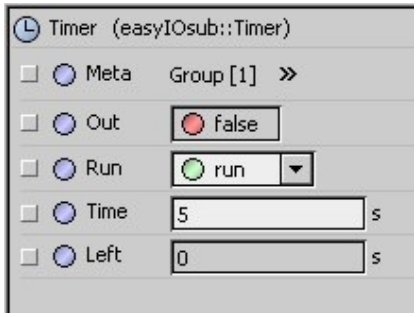
An example of TimeAvg object computing the average from Carbon Dioxide Value over period of every 1000ms.

24.5 *Timer*

Timer outputs a pulse for the configured amount of time "in" is used to fire the timer:
- if low, out is forced to false
- if high, out = 1 until timer reaches "time" seconds

Alternatively, the pulse can be fired from the "Start Timer" action if in is not linked.

The property sheet of the object is shown below

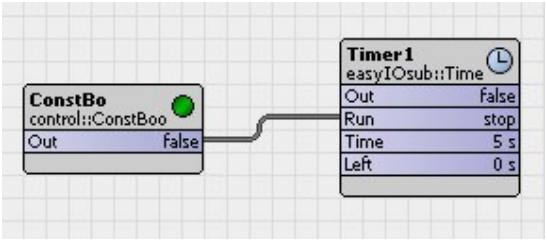


- **Out**
  A timed pulse output.

- **Run**
  Used to fire the timer on transition from false -> true

- **Time**
  Desire duration of the output pulse.

- **Left**
  Remaining time before the output transition from true -> false



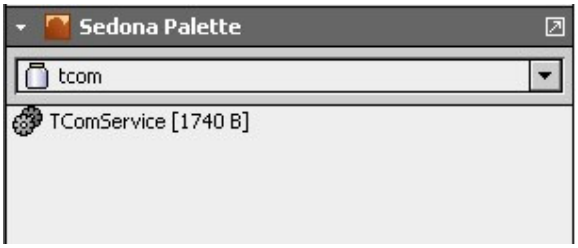
An example of Timer object hold the out value to true until the timer time count down to zero.

25 EasyioTcom

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
23	EasyioTcom	1.0.45.2	Easyio 1.0.43 or higher	TcomService

This kit contains 1 object. The objects are to be used for engineer the Sedona apps. This tcom service is for sedona protocol integration with Infocon TCom Driver.

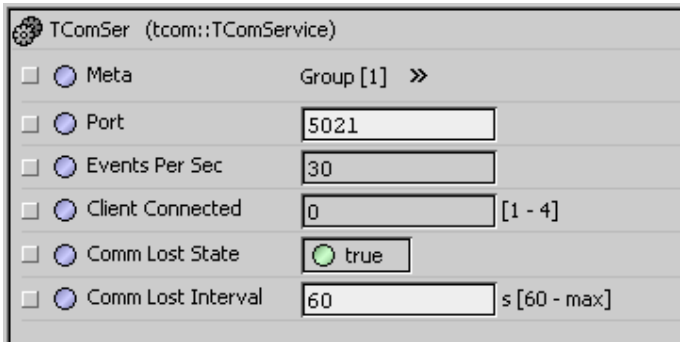
To use these objects just drag and drop into the wire sheet.



25.1 *TcomService*

TomService is an object use to integrate a sedona controller with Niagara station via sedona protocol with Infocon Tcom driver.

The property sheet of the object is shown below



- **Port**
  Port that uses to communicate between Niagara Station and a sedona controller. By default is “5021”. It is user editable.

- **Events Per Sec**
  A read only property. It will only handles 30 events per second and it is queue if there are more events. This will minimize the tcom driver load and controller load during the integration.

- **Client Connected**

A read only property.
Show total number of client connected to the Sedona controller.

- **Comm lost state**
  This property is a Boolean property where it monitor the tcom protocol time out. This property can be used as comm. monitoring where user can switch to local standalone schedule whenever the sedona controller comm. time out.

- **Com Lost Interval**
  This is the comm. monitoring interval check.
  By default is 60 seconds.

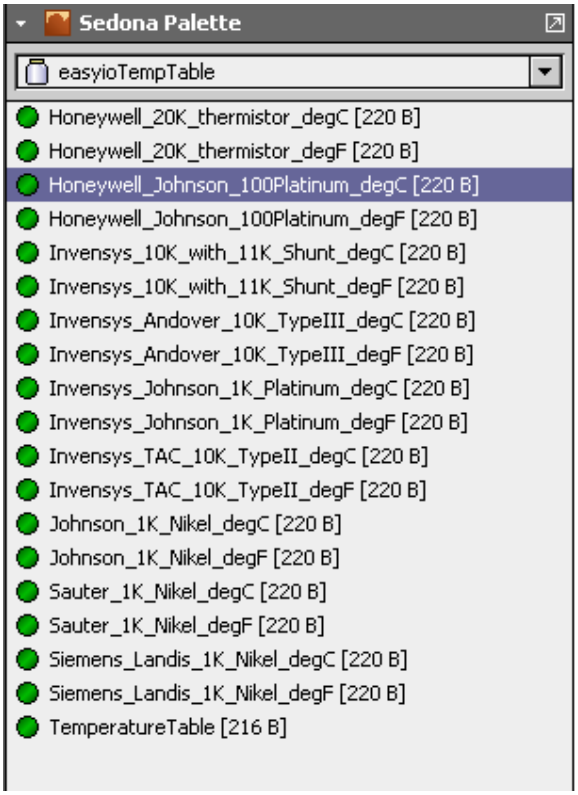
26 EasyioTempTable

Number	EasyIO Sedona Kit	Current Version	Dependencies	Components
24	EasyioTempTable	1.0.45.22	Easyio 1.0.43.10 or higher Platform Easyio 1.0.45.20	Honeywell / Johnson Pt100 Platinum (C) Honeywell / Johnson Pt100 Platinum (F) Honeywell 20K Thermistor (C) Honeywell 20K Thermistor (F) Invensys 10K with 11K shunt (C) Invensys 10K with 11K shunt (F) Invensys / Andover 10K Thermistor Type III (C) Invensys / Andover 10K Thermistor Type III (F) Invensys / Johnson Pt1000 Platinum (C) Invensys / Johnson Pt1000 Platinum (F) Invensys TAC 10K Thermistor Type II (C) Invensys TAC 10K Thermistor Type II (C) Johnson 1K Nikel (C) Johnson 1K Nikel (F) Sauter 1K Nikel (C) Sauter 1K Nikel (F) Siemens / Landis 1K Nikel (C) Siemens / Landis 1K Nikel (F) Temperature Table

This kit contains 19 objects. The object is to configure the temp table via Sedona workbench, without configuring via the web browser. This eliminates extra tools for configuring and backup.

These objects are also good for multiple typical controller duplication. No more extra tools for temperature table configuring.

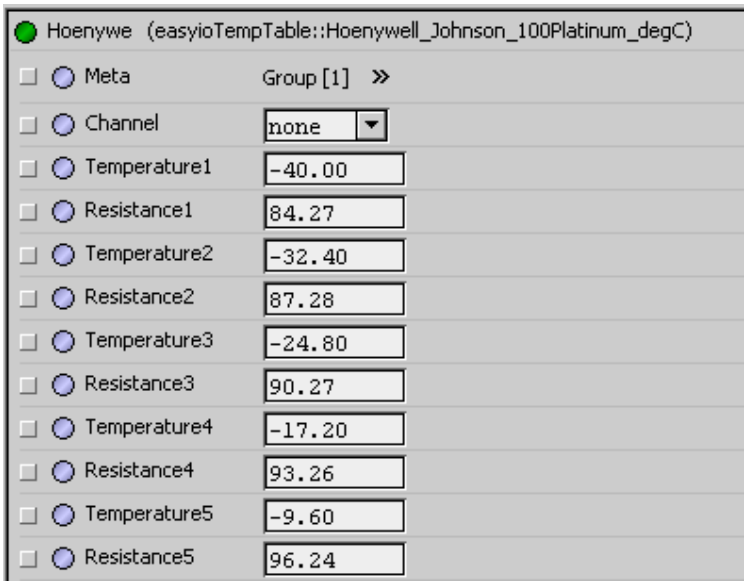
To use these objects just drag and drop into the wire sheet.



26.1 *Honeywell / Johnson Pt100 Platinum , (Celcius)*

Honeywell / Johnson Pt100 Platinum an extension object to the Temperature Table object. This object values are preset to commonly used Honeywell or Johnson Controls Pt100 Platinum Temperature Sensor resistance table versus Temperature Value in **Celsius**.
The temperature range for this sensor is from **-40 C° to 120 C°.**

The property sheet of the object is shown below

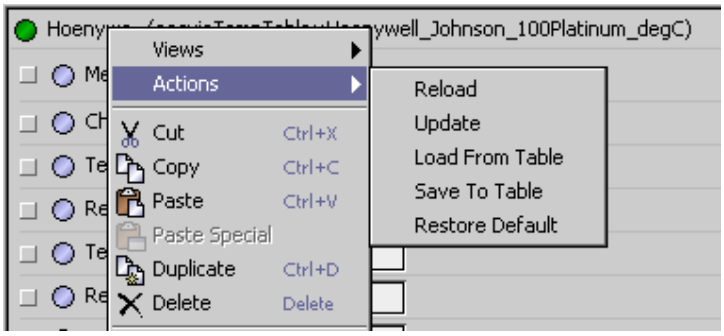


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions are available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

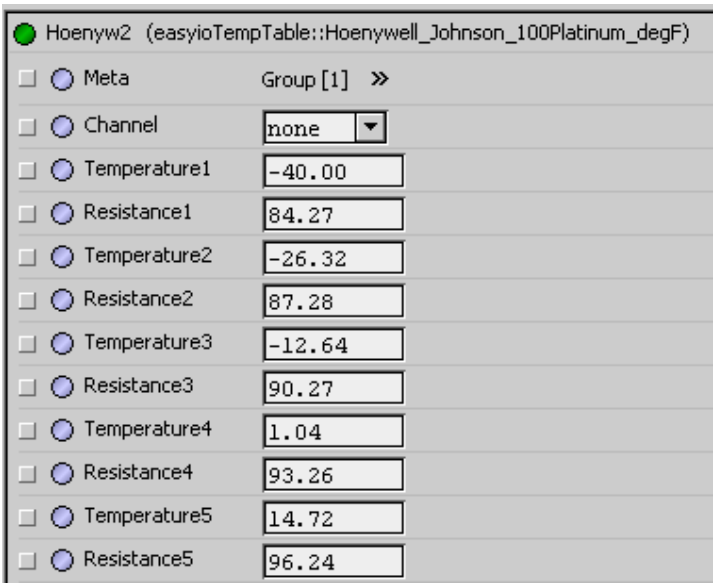
Table below shows the resistance value versus Temperature value

Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
84.27	-40.00	-40.00
92.16	-20.00	-4.00
96.08	-10.00	14.00
100.00	0.00	30.20
101.95	5.00	41.00
103.90	10.00	50.00
105.85	15.00	59.00
107.79	20.00	68.00
109.73	25.00	77.00
111.67	30.00	86.00
113.60	35.00	95.00
115.54	40..00	104.00
117.47	45.00	113.00
119.40	50.00	122.00
121.32	55.00	131.00
123.23	60.00	140.00
125.15	65.00	149.00
127.07	70.00	158.00
128.98	75.00	167.00
130.89	80.00	176.00
138.5	100.00	212.00
146.06	120.00	248.00

26.2 *Honeywell / Johnson Pt100 Platinum , (Fahrenheit)*

Honeywell / Johnson Pt100 Platinum is an extension object to the Temperature Table object. This object values are preset to commonly used Honeywell or Johnson Controls Pt100 Platinum Sensor resistance table versus Temperature Value in **Fahrenheit**.
The temperature range for this sensor is from **-40 F° to 248 F° .**

The property sheet of the object is shown below

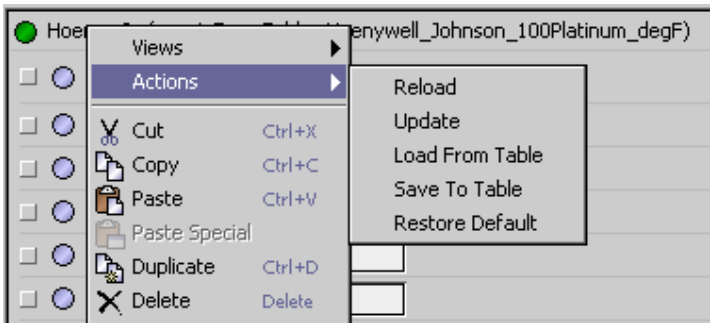


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

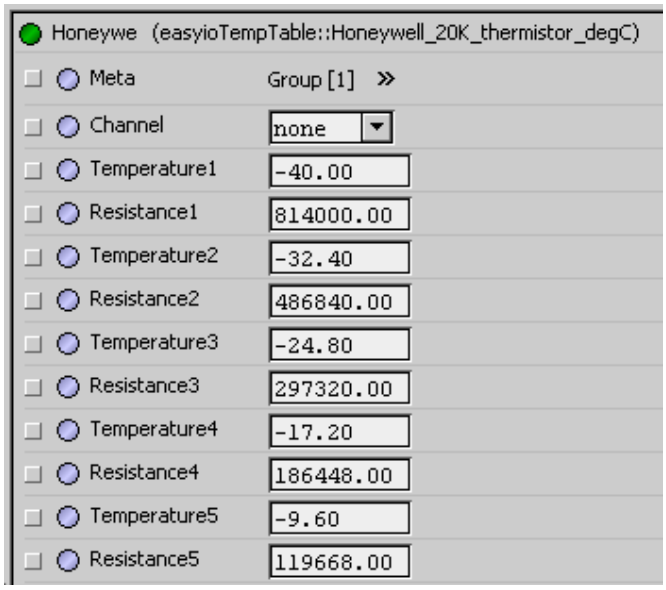
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
84.27	-40.00	-40.00
92.16	-20.00	-4.00
96.08	-10.00	14.00
100.00	0.00	30.20
101.95	5.00	41.00
103.90	10.00	50.00
105.85	15.00	59.00
107.79	20.00	68.00
109.73	25.00	77.00
111.67	30.00	86.00
113.60	35.00	95.00
115.54	40..00	104.00
117.47	45.00	113.00
119.40	50.00	122.00
121.32	55.00	131.00
123.23	60.00	140.00
125.15	65.00	149.00
127.07	70.00	158.00
128.98	75.00	167.00
130.89	80.00	176.00
138.5	100.00	212.00
146.06	120.00	248.00

26.3 *Honeywell 20K Thermistor , (Celcius)*

Honeywell 20K Thermistor is an extension object to the Temperature Table object. This object values are preset to commonly used Honeywell 20K Thermistor Sensor resistance table versus Temperature Value in **Celsius**.

The temperature range for this sensor is from **-40 C° to 120 C° .**

The property sheet of the object is shown below

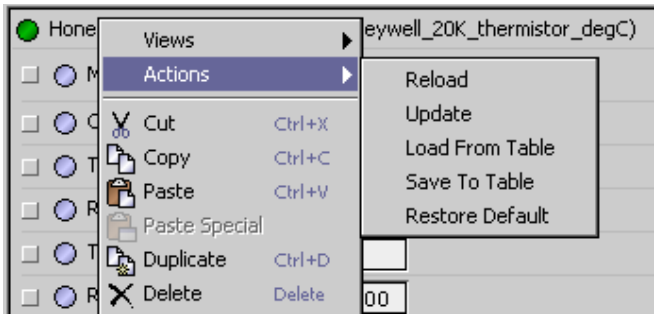


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:

Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

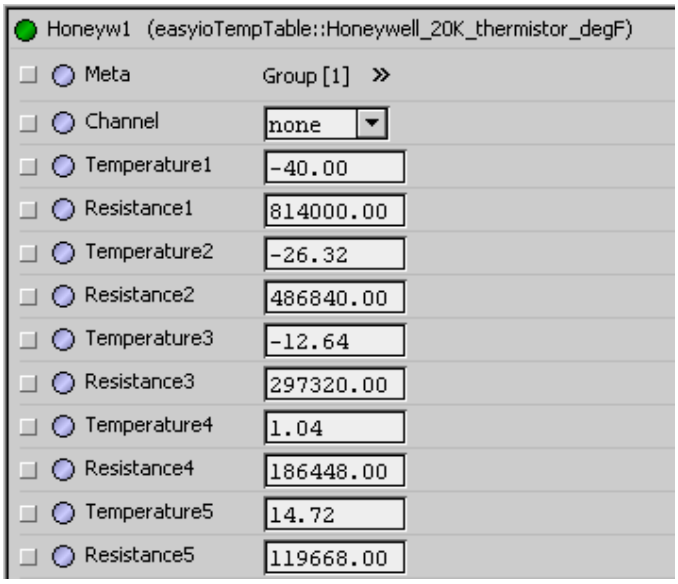
Table below shows the resistance value versus Temperature value

Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
814,000	-40.00	-40.00
220,060	-20.00	-4.00
122,380	-10.00	14.00
70,200	0.00	30.20
53,800	5.00	41.00
41,560	10.00	50.00
32,340	15.00	59.00
25,340	20.00	68.00
20,000	25.00	77.00
15,884	30.00	86.00
12,696	35.00	95.00
10,210	40..00	104.00
8,258	45.00	113.00
6,718	50.00	122.00
5,494	55.00	131.00
4,518	60.00	140.00
3,734	65.00	149.00
3,100	70.00	158.00
2,586	75.00	167.00
2,168	80.00	176.00
1,113.8	100.00	212.00
609	120.00	248.00

26.4 *Honeywell 20K Thermistor , (Fahrenheit)*

Honeywell 20K Thermistor is an extension object to the Temperature Table object. This object values are preset to commonly used Honeywell 20K Thermistor Sensor resistance table versus Temperature Value in **Fahrenheit.**
The temperature range for this sensor is from **-40 F° to 248 F° .**

The property sheet of the object is shown below

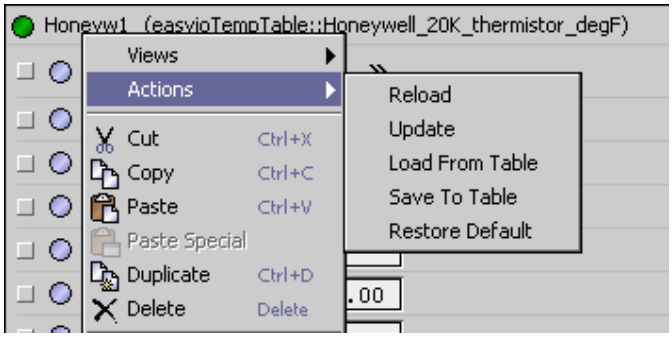


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:

Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

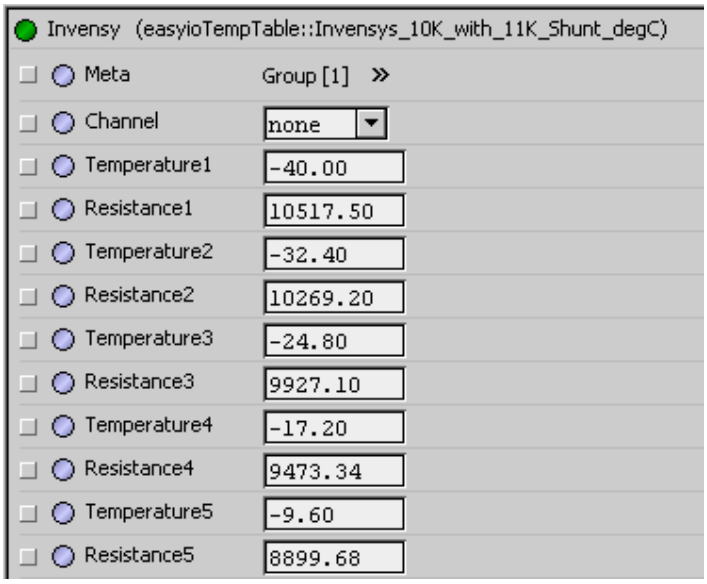
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
814,000	-40.00	-40.00
220,060	-20.00	-4.00
122,380	-10.00	14.00
70,200	0.00	30.20
53,800	5.00	41.00
41,560	10.00	50.00
32,340	15.00	59.00
25,340	20.00	68.00
20,000	25.00	77.00
15,884	30.00	86.00
12,696	35.00	95.00
10,210	40..00	104.00
8,258	45.00	113.00
6,718	50.00	122.00
5,494	55.00	131.00
4,518	60.00	140.00
3,734	65.00	149.00
3,100	70.00	158.00
2,586	75.00	167.00
2,168	80.00	176.00
1,113.8	100.00	212.00
609	120.00	248.00

26.5 *Invensys 10K Thermistor with 11K Shunt , (Celcius)*

Invensys 10K Thermistor with 11K Shunt is an extension object to the Temperature Table object. This object values are preset to commonly used Invensys 10K Thermistor with 11K Shunt Sensor resistance table versus Temperature Value in **Celsius**.

The temperature range for this sensor is from **-40 C° to120 C° .**

The property sheet of the object is shown below

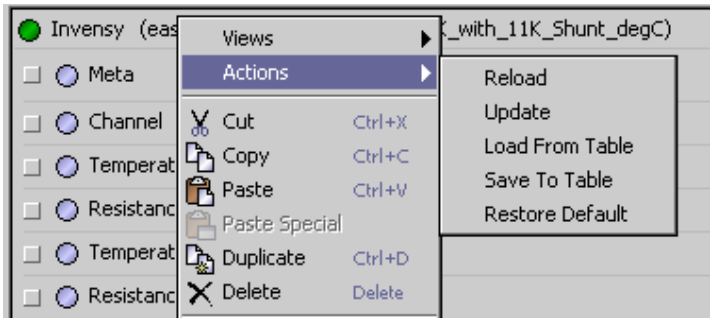


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:

Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

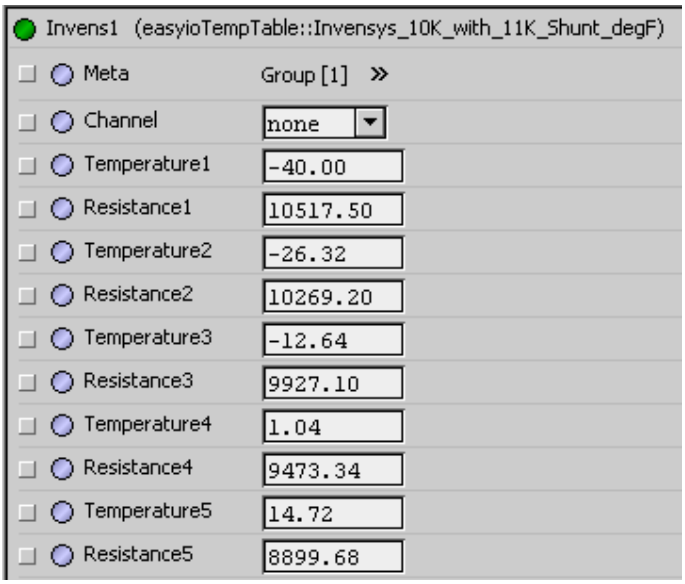
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
10,517.5	-40.00	-40.00
9,654.20	-20.00	-4.00
8,933.00	-10.00	14.00
8,011.60	0.00	30.20
7,488.70	5.00	41.00
6,938.20	10.00	50.00
6,369.30	15.00	59.00
5,797.90	20.00	68.00
5,238.10	25.00	77.00
4,695.90	30.00	86.00
4,183.90	35.00	95.00
3,707.30	40..00	104.00
3,270.80	45.00	113.00
2,875.40	50.00	122.00
2,520.70	55.00	131.00
2,206.40	60.00	140.00
1,928.90	65.00	149.00
1,685.10	70.00	158.00
1,472.40	75.00	167.00
1,287.40	80.00	176.00
760.30	100.00	212.00
461.60	120.00	248.00

26.6 *Invensys 10K Thermistor with 11K Shunt , (Fahrenheit)*

Invensys 10K Thermistor with 11K Shunt is an extension object to the Temperature Table object. This object values are preset to commonly used Invensys 10K Thermistor with 11K Shunt Sensor resistance table versus Temperature Value in **Fahrenheit**.

The temperature range for this sensor is from **-40 F° to 248 F° .**

The property sheet of the object is shown below

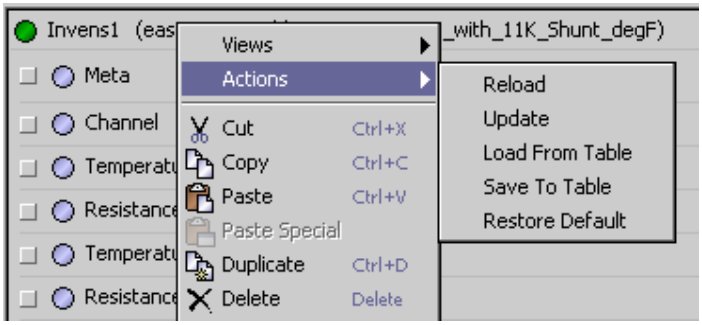


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:

Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

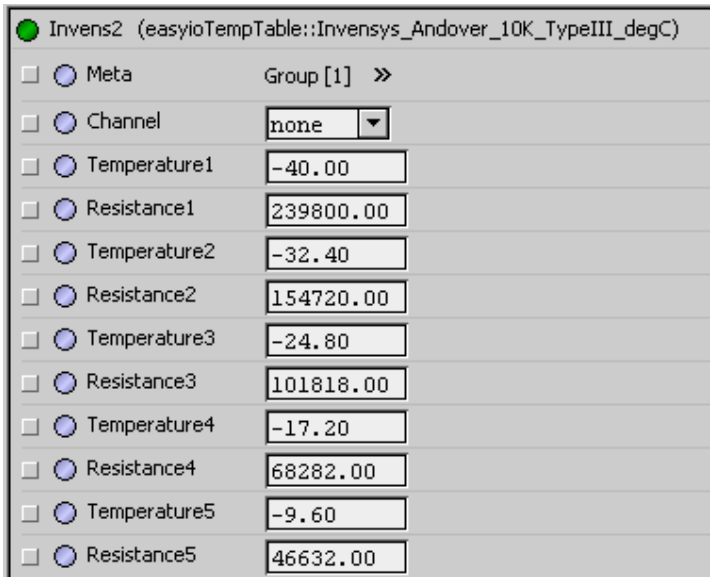
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
10517.5	-40.00	-40.00
9654.20	-20.00	-4.00
8933.00	-10.00	14.00
8011.60	0.00	30.20
7488.70	5.00	41.00
6938.20	10.00	50.00
6369.30	15.00	59.00
5797.90	20.00	68.00
5238.10	25.00	77.00
4695.90	30.00	86.00
4183.90	35.00	95.00
3707.30	40..00	104.00
3270.80	45.00	113.00
2875.40	50.00	122.00
2520.70	55.00	131.00
2206.40	60.00	140.00
1928.90	65.00	149.00
1685.10	70.00	158.00
1472.40	75.00	167.00
1287.40	80.00	176.00
760.30	100.00	212.00
461.60	120.00	248.00

26.7 *Invensys / Andover 10K Thermistor Type III , (Celcius)*

Invensys 10K Thermistor type III is an extension object to the Temperature Table object. This object values are preset commonly used Invensys or Andover 10K Thermistor Type III Sensor resistance table versus Temperature Value in **Celsius.**

The temperature range for this sensor is from **-40 C° to 120 C° .**

The property sheet of the object is shown below

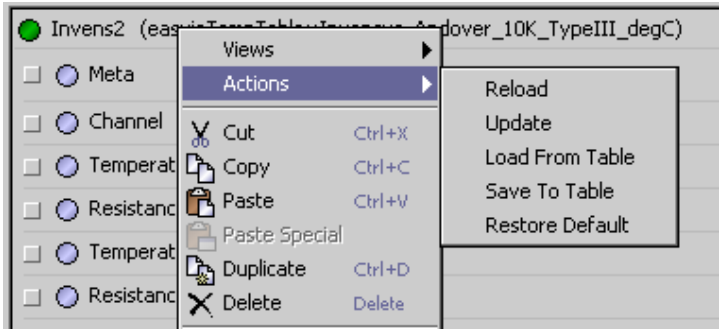


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

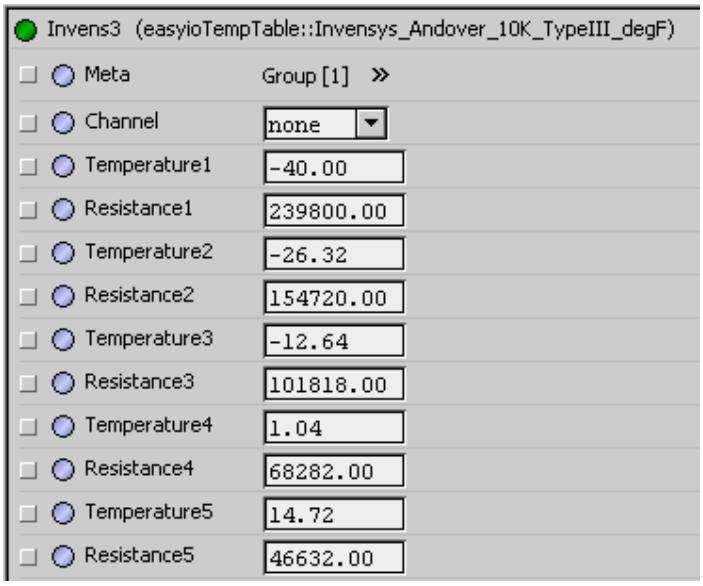
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
239,800	-40.00	-40.00
78,910	-20.00	-4.00
47,540	-10.00	14.00
29,490	0.00	30.20
23,460	5.00	41.00
18,790	10.00	50.00
15,130	15.00	59.00
12,260	20.00	68.00
10,000	25.00	77.00
8,194	30.00	86.00
6,752	35.00	95.00
5,592	40..00	104.00
4,655	45.00	113.00
3,893	50.00	122.00
3,270	55.00	131.00
2,760	60.00	140.00
2,339	65.00	149.00
1,990	70.00	158.00
1,700	75.00	167.00
1,458	80.00	176.00
816.8	100.00	212.00
481.8	120.00	248.00

26.8 *Invensys / Andover 10K Thermistor Type III , (Fahrenheit)*

Invensys 10K Thermistor type III is an extension object to the Temperature Table object. This object values are preset commonly used Invensys or Andover 10K Thermistor Type III Sensor resistance table versus Temperature Value in **Fahrenheit**.

The temperature range for this sensor is from **-40F° to 248F° .**

The property sheet of the object is shown below

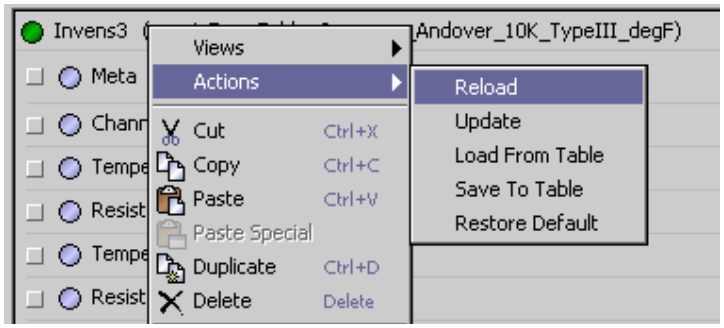


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

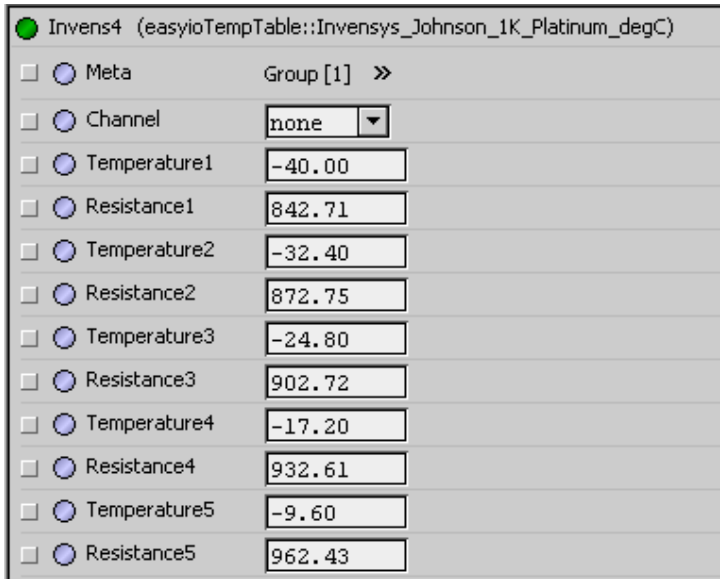
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
239,800	-40.00	-40.00
78,910	-20.00	-4.00
47,540	-10.00	14.00
29,490	0.00	30.20
23,460	5.00	41.00
18,790	10.00	50.00
15,130	15.00	59.00
12,260	20.00	68.00
10,000	25.00	77.00
8,194	30.00	86.00
6,752	35.00	95.00
5,592	40..00	104.00
4,655	45.00	113.00
3,893	50.00	122.00
3,270	55.00	131.00
2,760	60.00	140.00
2,339	65.00	149.00
1,990	70.00	158.00
1,700	75.00	167.00
1,458	80.00	176.00
816.8	100.00	212.00
481.8	120.00	248.00

26.9 *Invensys / Johnson Pt1000 Platinum , (Celcius)*

Invensys / Johnson Pt1000 Platinum is an extension object to the Temperature Table object. This object values are preset to commonly used Invensys or Johnson Pt1000 Platinum Sensor resistance table versus Temperature Value in **Celsius.**

The temperature range for this sensor is from **-40 C° to 120 C°.**

The property sheet of the object is shown below

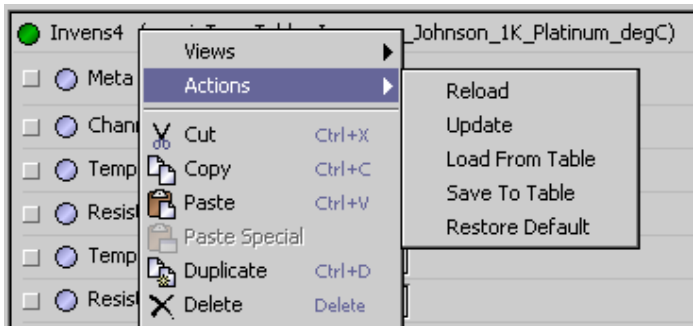


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
842.71	-40.00	-40.00
921.60	-20.00	-4.00
960.86	-10.00	14.00
1000.00	0.00	30.20
1019.53	5.00	41.00
1039.02	10.00	50.00
1058.50	15.00	59.00
1077.93	20.00	68.00
1097.34	25.00	77.00
1116.72	30.00	86.00
1136.07	35.00	95.00
1155.39	40..00	104.00
1174.68	45.00	113.00
1193.95	50.00	122.00
1213.19	55.00	131.00
1232.39	60.00	140.00
1251.57	65.00	149.00
1270.72	70.00	158.00
1289.83	75.00	167.00
1308.93	80.00	176.00
1285.00	100.00	212.00
1460.61	120.00	248.00

26.10 *Invensys / Johnson Pt1000 Platinum , (Fahrenheit)*

Invensys / Johnson Pt1000 Platinum is an extension object to the Temperature Table object. This object values are preset to commonly used Invensys or Johnson Pt1000 Platinum Sensor resistance table versus Temperature Value in **Fahrenheit.**

The temperature range for this sensor is from **-40 F° to248 F° .**

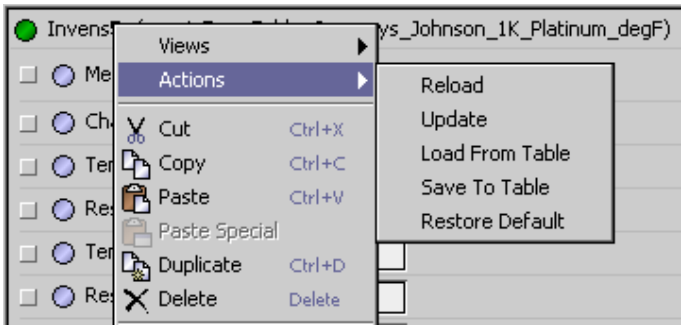
The property sheet of the object is shown below

- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:

Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

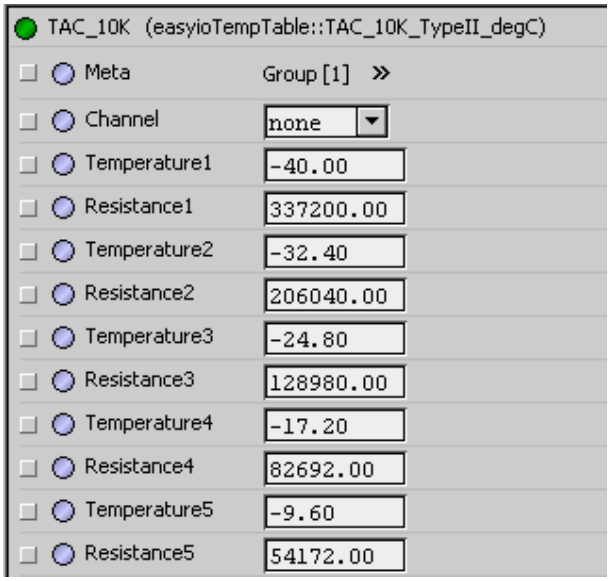
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
842.71	-40.00	-40.00
921.60	-20.00	-4.00
960.86	-10.00	14.00
1000.00	0.00	30.20
1019.53	5.00	41.00
1039.02	10.00	50.00
1058.50	15.00	59.00
1077.93	20.00	68.00
1097.34	25.00	77.00
1116.72	30.00	86.00
1136.07	35.00	95.00
1155.39	40..00	104.00
1174.68	45.00	113.00
1193.95	50.00	122.00
1213.19	55.00	131.00
1232.39	60.00	140.00
1251.57	65.00	149.00
1270.72	70.00	158.00
1289.83	75.00	167.00
1308.93	80.00	176.00
1285.00	100.00	212.00
1460.61	120.00	248.00

26.11 *Invensys / TAC 10K Thermistor Type II , (Celcius)*

Invensys / TAC 10K Thermistor Type II is an extension object to the Temperature Table object. This object values are preset to commonly used Invensys 10K Thermistor Type II Sensor resistance table versus Temperature Value in **Celsius.**

The temperature range for this sensor is from **-40 C° to120 C° .**

The property sheet of the object is shown below

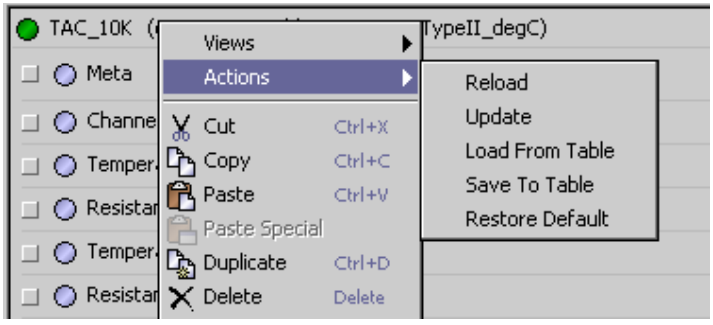


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:

Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

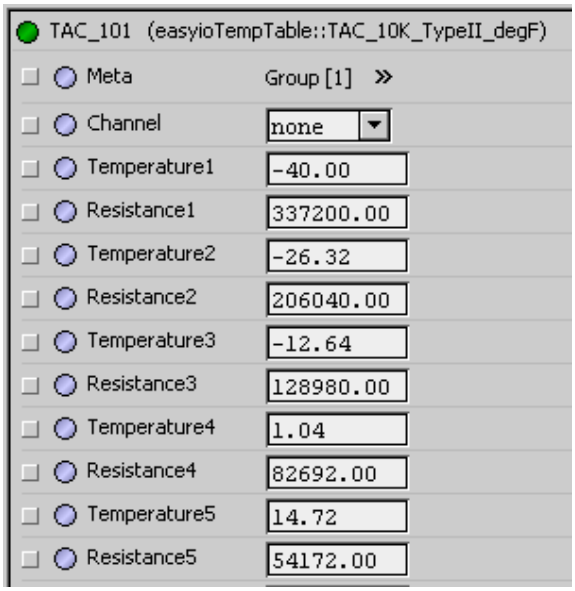
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
337200	-40.00	-40.00
97130	-20.00	-4.00
55340	-10.00	14.00
32660	0.00	30.20
25400	5.00	41.00
19900	10.00	50.00
15710	15.00	59.00
12490	20.00	68.00
10000	25.00	77.00
8056	30.00	86.00
6531	35.00	95.00
5326	40..00	104.00
4368	45.00	113.00
3602	50.00	122.00
2987	55.00	131.00
2489	60.00	140.00
2084	65.00	149.00
1753	70.00	158.00
1482	75.00	167.00
1258	80.00	176.00
679.8	100.00	212.00
389.4	120.00	248.00

26.12 *Invensys / TAC 10K Thermistor Type II , (Fahrenheit)*

Invensys / TAC 10K Thermistor Type II is an extension object to the Temperature Table object. This object values are preset to commonly used Invensys 10K Thermistor Type II Sensor resistance table versus Temperature Value in **Fahrenheit.**

The temperature range for this sensor is from **-40 F° to 248 F° .**

The property sheet of the object is shown below

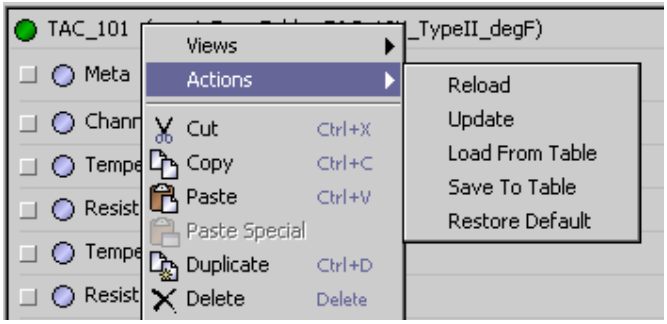


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

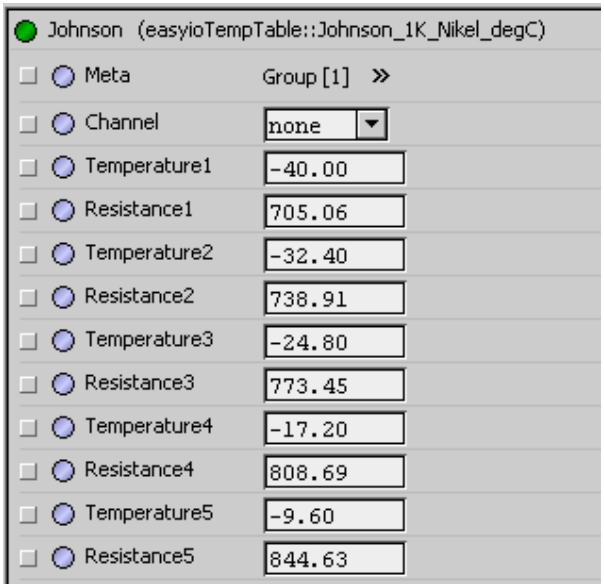
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
337200	-40.00	-40.00
97130	-20.00	-4.00
55340	-10.00	14.00
32660	0.00	30.20
25400	5.00	41.00
19900	10.00	50.00
15710	15.00	59.00
12490	20.00	68.00
10000	25.00	77.00
8056	30.00	86.00
6531	35.00	95.00
5326	40..00	104.00
4368	45.00	113.00
3602	50.00	122.00
2987	55.00	131.00
2489	60.00	140.00
2084	65.00	149.00
1753	70.00	158.00
1482	75.00	167.00
1258	80.00	176.00
679.8	100.00	212.00
389.4	120.00	248.00

26.13 *Johnson 1K Nikel , (Celcius)*

Johnson Control 1K Nikel is an extension object to the Temperature Table object. This object values are preset to commonly used Johnson Control 1K Nikel Sensor resistance table versus Temperature Value in **Celsius.**

The temperature range for this sensor is from **-40 C° to120 C° .**

The property sheet of the object is shown below

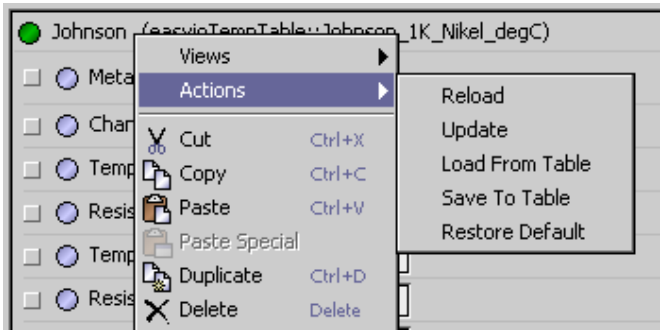


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:

Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
705.06	-40.00	-40.00
795.63	-20.00	-4.00
842.72	-10.00	14.00
891.00	0.00	30.20
915.58	5.00	41.00
940.46	10.00	50.00
965.64	15.00	59.00
991.12	20.00	68.00
1016.89	25.00	77.00
1042.97	30.00	86.00
1069.35	35.00	95.00
1096.03	40..00	104.00
1123.02	45.00	113.00
1150.33	50.00	122.00
1177.95	55.00	131.00
1205.88	60.00	140.00
1234.15	65.00	149.00
1262.73	70.00	158.00
1291.66	75.00	167.00
1320.92	80.00	176.00
1441.48	100.00	212.00
1568.01	120.00	248.00

26.14 *Johnson 1K Nikel , (Fahrenheit)*

Johnson Control 1K Nikel is an extension object to the Temperature Table object. This object values are preset to commonly used Johnson Control 1K Nikel Sensor resistance table versus Temperature Value in **Fahrenheit.**

The temperature range for this sensor is from **-40 F° to 248 F°.**

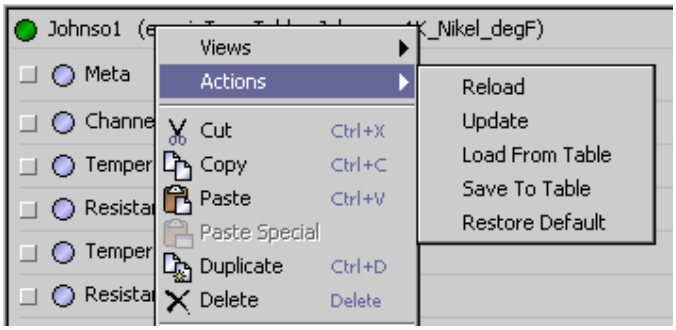
The property sheet of the object is shown below

- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

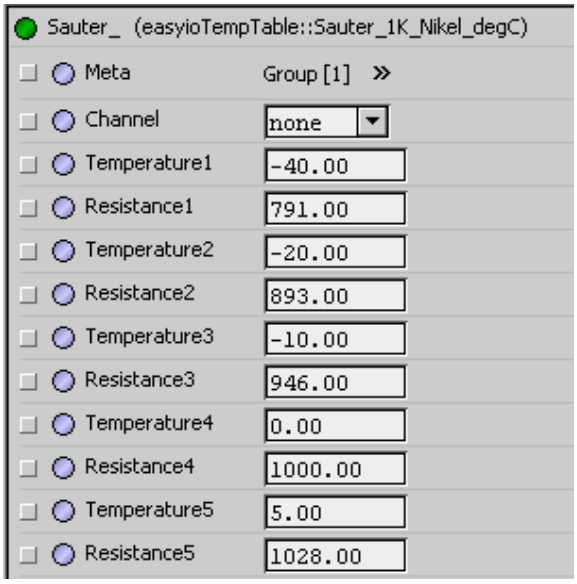
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
705.06	-40.00	-40.00
795.63	-20.00	-4.00
842.72	-10.00	14.00
891.00	0.00	30.20
915.58	5.00	41.00
940.46	10.00	50.00
965.64	15.00	59.00
991.12	20.00	68.00
1016.89	25.00	77.00
1042.97	30.00	86.00
1069.35	35.00	95.00
1096.03	40..00	104.00
1123.02	45.00	113.00
1150.33	50.00	122.00
1177.95	55.00	131.00
1205.88	60.00	140.00
1234.15	65.00	149.00
1262.73	70.00	158.00
1291.66	75.00	167.00
1320.92	80.00	176.00
1441.48	100.00	212.00
1568.01	120.00	248.00

26.15 *Sauter 1K Nikel , (Celcius)*

Sauter 1K Nikel is an extension object to the Temperature Table object. This object values are preset to commonly used Sauter 1K Nikel Sensor resistance table versus Temperature Value in **Celsius.**

The temperature range for this sensor is from **-40 C° to120 C° .**

The property sheet of the object is shown below

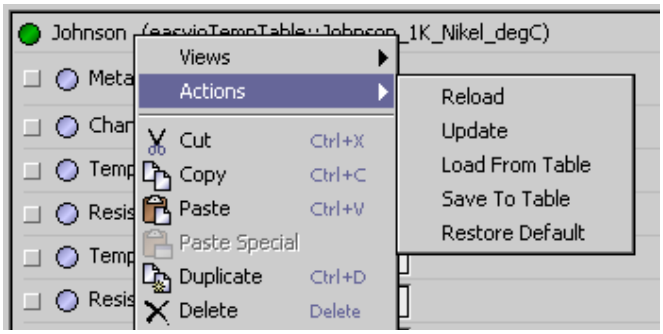


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:

Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

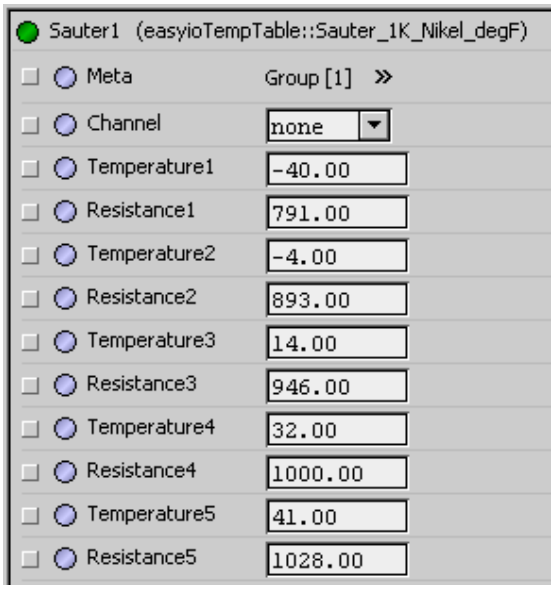
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
705.06	-40.00	-40.00
795.63	-20.00	-4.00
842.72	-10.00	14.00
891.00	0.00	30.20
915.58	5.00	41.00
940.46	10.00	50.00
965.64	15.00	59.00
991.12	20.00	68.00
1016.89	25.00	77.00
1042.97	30.00	86.00
1069.35	35.00	95.00
1096.03	40..00	104.00
1123.02	45.00	113.00
1150.33	50.00	122.00
1177.95	55.00	131.00
1205.88	60.00	140.00
1234.15	65.00	149.00
1262.73	70.00	158.00
1291.66	75.00	167.00
1320.92	80.00	176.00
1441.48	100.00	212.00
1568.01	120.00	248.00

26.16 *Sauter 1K Nikel , (Fahrenheit)*

Sauter 1K Nikel is an extension object to the Temperature Table object. This object values are preset to commonly used Sauter 1K Nikel Sensor resistance table versus Temperature Value in **Fahrenheit.**

The temperature range for this sensor is from **-40 F° to 248 F°.**

The property sheet of the object is shown below

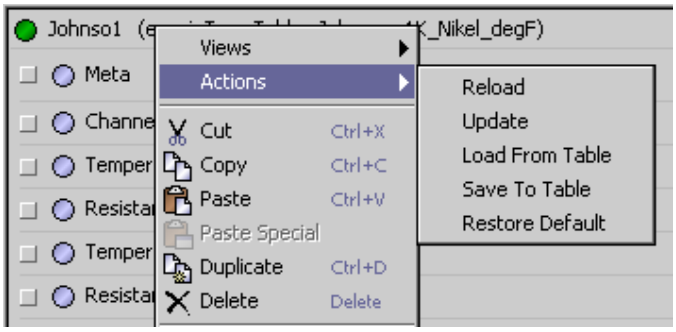


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

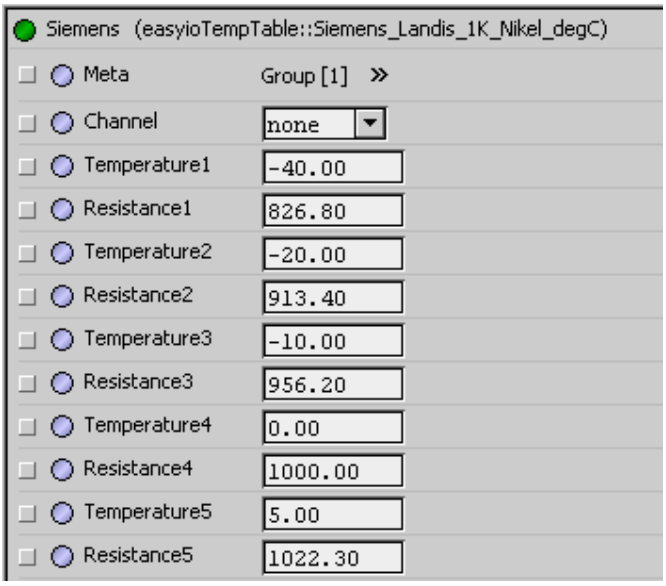
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
705.06	-40.00	-40.00
795.63	-20.00	-4.00
842.72	-10.00	14.00
891.00	0.00	30.20
915.58	5.00	41.00
940.46	10.00	50.00
965.64	15.00	59.00
991.12	20.00	68.00
1016.89	25.00	77.00
1042.97	30.00	86.00
1069.35	35.00	95.00
1096.03	40..00	104.00
1123.02	45.00	113.00
1150.33	50.00	122.00
1177.95	55.00	131.00
1205.88	60.00	140.00
1234.15	65.00	149.00
1262.73	70.00	158.00
1291.66	75.00	167.00
1320.92	80.00	176.00
1441.48	100.00	212.00
1568.01	120.00	248.00

26.17 *Siemens/Landis 1K Nikel , (Celcius)*

Siemens/Landis 1K Nikel is an extension object to the Temperature Table object. This object values are preset to commonly used Siemens/Landis 1K Nikel Sensor resistance table versus Temperature Value in **Fahrenheit.**

The temperature range for this sensor is from **-40 C° to 120 C°.**

The property sheet of the object is shown below

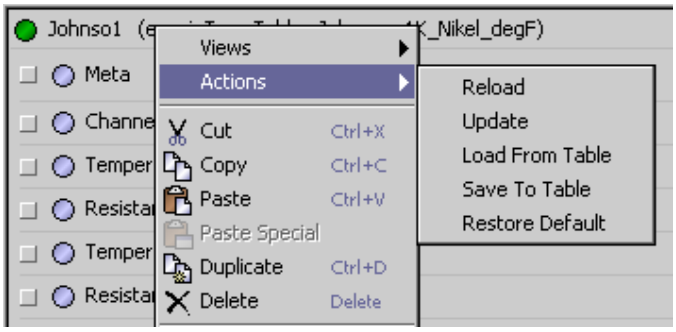


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

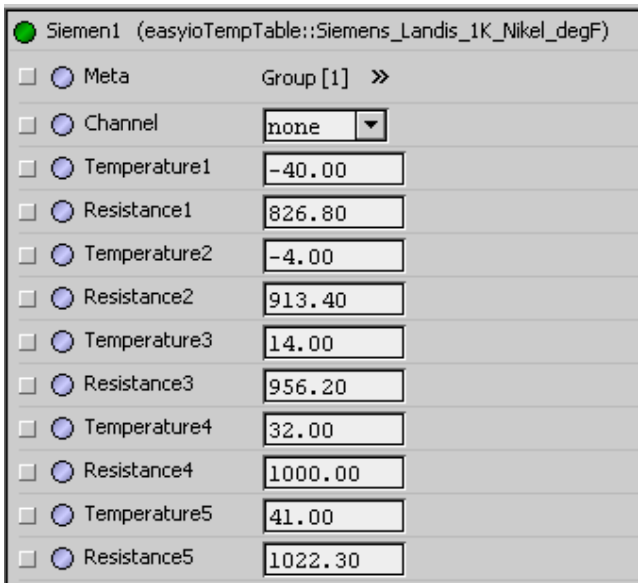
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
826.80	-40.00	-40.00
913.40	-20.00	-4.00
956.20	-10.00	14.00
1000.00	0.00	30.20
1022.30	5.00	41.00
1044.80	10.00	50.00
1067.60	15.00	59.00
1090.70	20.00	68.00
1114.00	25.00	77.00
1137.60	30.00	86.00
1161.50	35.00	95.00
1185.70	40..00	104.00
1210.20	45.00	113.00
1235.00	50.00	122.00
1260.10	55.00	131.00
1285.40	60.00	140.00
1311.10	65.00	149.00
1337.10	70.00	158.00
1363.50	75.00	167.00
1390.10	80.00	176.00
1500.00	100.00	212.00
1625.40	120.00	248.00

26.18 *Siemens/Landis 1K Nikel , (Fahrenheit)*

Siemens/Landis 1K Nikel is an extension object to the Temperature Table object. This object values are preset to commonly used Siemens/Landis 1K Nikel Sensor resistance table versus Temperature Value in **Fahrenheit.**

The temperature range for this sensor is from **-40 F° to 248 F°.**

The property sheet of the object is shown below

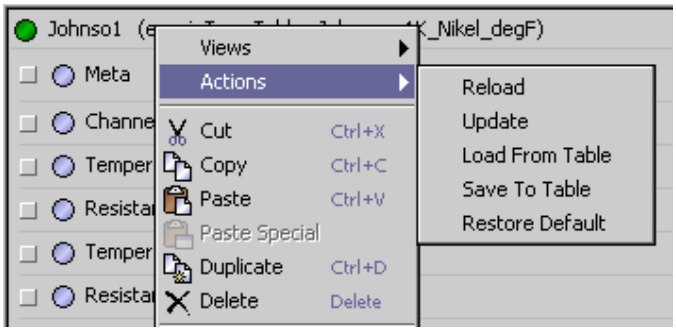


- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions is available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table or the last save table in non-volatile memory.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory

Restore Default :
To restore the table values back to factory setting as per table below.

Table below shows the resistance value versus Temperature value

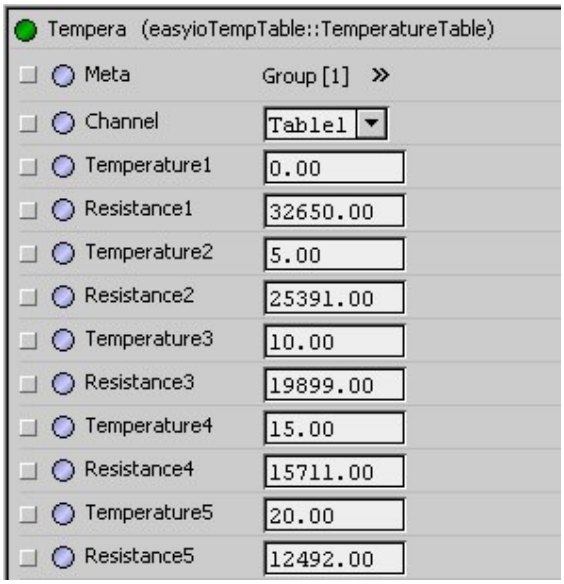
Resistance Value (Ohm)	Temp Value (Celcius)	Temp Value (Fahrenheit)
826.80	-40.00	-40.00
913.40	-20.00	-4.00
956.20	-10.00	14.00
1000.00	0.00	30.20
1022.30	5.00	41.00
1044.80	10.00	50.00
1067.60	15.00	59.00
1090.70	20.00	68.00
1114.00	25.00	77.00
1137.60	30.00	86.00
1161.50	35.00	95.00
1185.70	40..00	104.00
1210.20	45.00	113.00
1235.00	50.00	122.00
1260.10	55.00	131.00
1285.40	60.00	140.00
1311.10	65.00	149.00
1337.10	70.00	158.00
1363.50	75.00	167.00
1390.10	80.00	176.00
1500.00	100.00	212.00
1625.40	120.00	248.00

26.19 *Temperature Table*

TemperatureTable is an object for user to define the temperature sensor table resistance value versus the temperature value according to the manufacturer specifications.

User may use this object if any of the preset objects in this kit is not suitable.

The property sheet of the object is shown below



- **Channel**
  Channel is referring to the temp table from number 1 to 8 where it is user customizable.

  Channel 9 to 16 is predefine table and is not editable.
  Channel 9 to 16 is preset to temp table as below image.

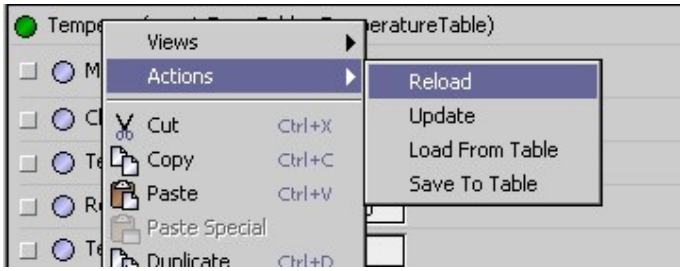
Analog Input Temperature Table Selection

Temperature Table Index	Type of Temperature Sensor
1 – 8	User defined Table 1 – 8 (default = table 9 – 16)
9	10K shunt (11K) Thermistor in Degree C
10	10K Thermistor in Degree C
11	1K Balco in Degree C
12	1K Platinum in Degree C
13	10K shunt (11K) Thermistor in Degree Fahrenheit
14	10K Thermistor in Degree Fahrenheit
15	1K Balco in Degree Fahrenheit
16	1K Platinum in Degree Fahrenheit

- **Temperature**
  Temperature Value that will be tie back to the resistance Value. (1-22 Lines)

- **Resistance**
  Resistance Value that will te back to the temperature value (1-22 Lines)

- **Actions**
  Actions are available when right mouse button at the object. It will show as below:



Reload:
Reload is loading from current selection table channel, fast action

Update:
Update is just update to the EasyIO Temperature table, this table will be update to the controller as long as the temperature table kits object is in the apps. Once the kit is deleted, the temperature table will be restored back with the original table.

Load from Table:
Load loading from one of the exiting available table, 1 - 16

Save to Table:
Save to table is permanently save to non-volatile memory